T.C.
Uludağ Üniversitesi
Fen Bilimleri Enstitüsü

EVRİMSEL ALGORİTMALARLA FİLTRE TASARIMLARI

Yiğit Çağatay KUYU

Yüksek Lisans Tezi

EVRİMSEL ALGORİTMALARLA FİLTRE TASARIMLARI

Yiğit Çağatay KUYU

T.C.
ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EVRİMSEL ALGORİTMALARLA FİLTRE TASARIMLARI

Yiğit Çağatay KUYU

Doç. Dr. Fahri VATANSEVER
(Danışman)

YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2016

TEZ ONAYI

Yiğit Çağatay Kuyu tarafından hazırlanan "Evrimsel Algoritmalarla Filtre Tasarımları" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Uludağ Üniversitesi Fen Bilimleri Enstitüsü Elektronik Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Doç. Dr. Fahri VATANSEVER

Başkan : Doç. Dr. Fahri VATANSEVER — İMZA
Uludağ Üniversitesi
Mühendislik Fakültesi
Devreler ve Sistemler Anabilim Dalı

Üye : Yrd. Doç. Dr. Sait Eser KARLIK — İMZA
Uludağ Üniversitesi
Mühendislik Fakültesi
Telekomünikasyon Anabilim Dalı

Üye : Yrd. Doç. Dr. Cemal HANİLÇİ — İMZA
Bursa Teknik Üniversitesi
Doğa Bilimleri, Mimarlık ve Mühendislik Fakültesi
Elektrik Elektronik Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. Ali Osman DEMİR
Enstitü Müdürü
29/7/2016

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

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,

- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,

- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,

- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,

- kullanılan verilerde herhangi bir tahrifat yapmadığımı,

- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

15/07/2016

Yiğit Çağatay Kuyu

ÖZET
Yüksek Lisans Tezi

EVRİMSEL ALGORİTMALARLA FİLTRE TASARIMLARI

Yiğit Çağatay KUYU

Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Fahri VATANSEVER

Elektrik-elektronik mühendisliği alanında filtreler önemli yer tutmaktadır. Filtre tasarımları için birçok yöntem mevcuttur. Bu tez çalışmasında literatürde sıklıkla kullanılan genetik, diferansiyel gelişim, parçacık sürü optimizasyon, karınca koloni optimizasyon, armoni arama algoritmalarının yanısıra son yıllarda geliştirilmiş algoritmalardan olan girdap arama ve geri izleme arama optimizasyon algoritmalarıyla analog ve sayısal filtre tasarımları gerçekleştirilmiştir. Her bir algoritmanın çalışma prensibi ve çözüme ulaşırken kullandığı metotlar açıklanmıştır.

Sonlu darbe cevaplı (FIR) ve sonsuz darbe cevaplı (IIR) sayısal filtrelerin matematiksel ifadeleri, tanımlamaları ve özellikleri anlatılmış, her bir filtre türü için iki farklı örnek problem ele alınmıştır. Bu örnek problemlerde, alçak ve yüksek geçiren filtrelerin optimal filtre katsayıları evrimsel algoritmalar aracılığıyla bulunmuş, algoritmalara ait hata performansları analiz edilmiş, tasarımların doğruluğu ilişkili algoritmanın frekans cevap grafiği ile gözlemlenmiştir.

Analog filtre tasarımlarında kullanılan topolojilerden bahsedilmiş, ilgili topolojilere ilişkin devre çizimleri ve bağıntılarına yer verilmiştir. Evrimsel algoritmalar vasıtasıyla alçak geçiren aktif analog filtre, durum değişkenli ve Sallen-Key filtre topolojileri kullanılarak tasarlanmıştır. Tasarımların gerçeklenmesini kolaylaştırmak açısından, algoritmalar tarafından bulunan devre elemanları endüstri üretim serilerinden seçilmiştir. Elde edilen devre elemanlarına göre, SPICE benzetim programı (PSpice) aracılığıyla devreler gerçeklenmiş ve ilgili benzetim grafikleriyle tasarlanan devreler doğrulanmıştır.

Anahtar Kelimeler: Analog filtre, sayısal filtre, evrimsel algoritma

2016, ix + 94 sayfa.

ABSTRACT
MSc Thesis
FILTER DESIGNS WITH EVOLUTIONARY ALGORITHMS

Yiğit Çağatay KUYU

Uludağ University
Graduate School of Natural and Applied Sciences
Department of Electronics Engineering

Supervisor: Assoc. Prof. Dr. Fahri VATANSEVER

Filters are the most important thing in the area of electrical-electronic engineering. There are various ways to design a filter. In this thesis, analog and digital filters were designed with genetic, differential evolution, particle swarm optimization, ant colony optimization, harmony search algorithms which are commonly used in the literature as well as vortex searh and backtracking search optimization algorithms developed in recent years. The working principle of each algorithm and the methods which are used to reach a solution were described.

Finite impulse response (FIR) and infinite impulse response (IIR) digital filters' mathematical equations and properties were explained, two different problems were taken as examples for each of the filter type. In these examples, low-pass and high-pass optimal filter cofficients were found through evolutionary algorithms, error performances were analyzed, the accuracy of the designs was validated with frequency response graph of the associated algorithm.

The topologies used in analog filter design were mentioned, circuit layouts and equations related to the topologies were given. Low-pass active analog filter was designed with evolutionary algorithms by using state variable and Sallen-Key topologies. Components' values found by the algorithms were selected from manufactured series in order to making realization of designs easily. According to components values obtained from the algorithms, the circuits were implemented with SPICE simulation program (Pspice) and the designed circuits were validated through related simulation graphs.

Key words: Analog filter, digital filter, evolutionary algorithm.

2016, ix + 94 pages.

ÖNSÖZ VE TEŞEKKÜR

Bu çalışmada bilgisi ve tecrübesi ile her türlü konuda yardımlarını benden esirgemeyen değerli danışmanım Sayın Doç. Dr. Fahri VATANSEVER'e teşekkürlerimi sunarım. Ayrıca tez çalışmam süresince maddi manevi desteklerini benden esirgemeyen değerli aileme ve Sayın Gözde IŞIK'a teşekkürü bir borç bilirim.

Yiğit Çağatay Kuyu

15/07/2016

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	**Açıklama**
Q	Kalite faktörü
ΔQ	Kalite faktörü sapması
Δω	Kesim frekansı sapması
F	Ölçekleme faktörü
H_i	Filtrenin ideal frekans cevabı
H_b	Filtrenin bulunan frekans cevabı

Kısaltmalar	**Açıklama**
ABC	Artificial Bee Colony Algorithm (Yapay Arı Kolonisi Algoritması)
ACO	Ant Colony Optimization Algorithm (Karınca Koloni Optimizasyon Algoritması)
CSA	Clonal Selection Algorithm (Klonal Seçim Algoritması)
DE	Differential Evolution Algorithm (Diferansiyel Gelişim Algoritması)
FIR	Finite Impulse Response Filter (Sonlu Dürtü Cevaplı Filtre)
GA	Genetic Algorithm (Genetik Algoritma)
HS	Harmony Search Algorithm (Armoni Arama Algoritması)
IIR	Infinite Impulse Response Filter (Sonsuz Dürtü cevaplı filtre)
MFB	Multiple Feedback Topology (Çoklu Geri Besleme Topoloji)
PSO	Particle Swarm Optimization Algorithm (Parçacık Sürü Optimizasyon Algoritması)
SV	State Variable Topology (Durum Değişkenli Topoloji)
VCVS	Voltage Controlled Voltage Source (Gerilim Kontrollü Gerilim Kaynağı)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

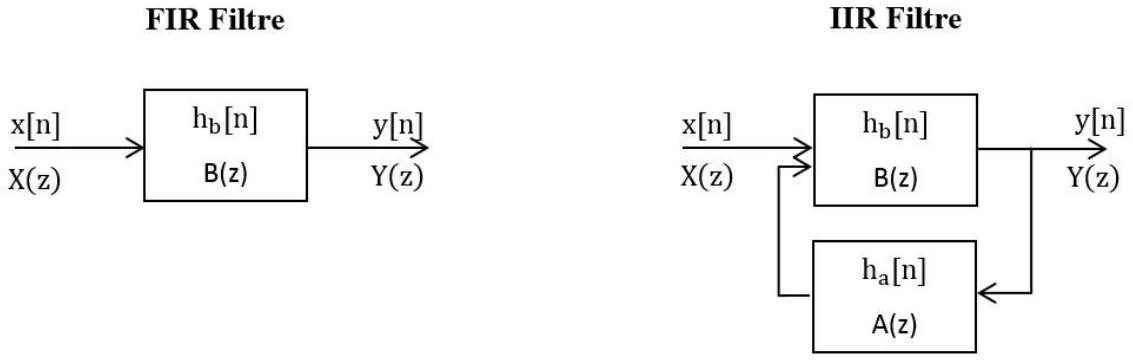
İşaretler fiziksel bir durum hakkında bilgi taşıyan, bir ya da birden fazla değişkene sahip fonksiyonlar olarak adlandırılırlar. İşaretlerin iletim ortamı yapısı sebebiyle veya çevresel etmenlerden dolayı üzerinde oluşan istenmeyen gürültü bileşenleri yapısını bozmakta ve bilgi taşıyan işareti sınırlandırmaktadır (Vaseghi 2009). Filtreler bazı işaretlerin geçmesini, bazı işaretlerin durdurulmasını, aynı ortamda bulunan birden fazla işaretin ayrıştırılmasını ve gürültü gibi istenmeyen etkilerin azaltmasını veya yok edilmesini sağlamak amacıyla kullanılmaktadırlar (Ertürk 2009). Filtreler analog ve sayısal olmak üzere ikiye ayrılmaktadırlar. Analog filtrelerin aktif ve pasif filtereler olmak üzere iki türü mevcuttur. Pasif filtreler direnç, kapasitör ve bobin (indüktör) gibi devre elemanları (pasif devre elemanları) bulunduran yapılardır. Bu pasif elemanlar bağlanış şekillerine göre belirli frekansları geçirip belirli frekansları sönümlendirebilmektedir. Aktif filtre devresinde pasif devre elemanlarına ek olarak transistor, bobin veya mikroişlemci gibi devre elemanları bulunabilmektedir. Bu tür devrelerde filtreleme işlemini yapan pasif devre elemanlarıdır (Winder 2002). Aktif filtrelerin kullanımda sağladığı temel avantajlar aşağıdaki gibi özetlenebilmektedir (Bakshi ve ark. 2011, Anonim 2012).

- Kaskat yapılar için son derece uygun olup yüksek dereceden filtreler düşük dereceden filtrelerin kaskat bağlanmasıyla tasarlanabilmektedir.
- Aktif filtre devrelerinin düşük çıkış empedansı ve yüksek giriş empedansına sahip olmasından dolayı girişine ve çıkışına bağlanacak devre veya devre elemanları arasında iyi bir izolasyon sağlamaktadır.
- Aktif elemanlar filtrenin geçirgen olduğu frekanslarda kuvvetlendirme sağlayacağından dolayı aktif filtreler kazanca sahiptirler.
- Aktif filtreler düşük frekanslı pasif filtreler için gerekli olan büyük indüktör ve kapasitörlere duyulan ihtiyacı ortadan kaldırdığından dolayı düşük maliyetle tasarlanabilmektedir.

Aktif filtreler yukarıdaki avantajlarından dolayı analog filtre tasarımlarında sıklıkla kullanılmaktadır. Analog filtre tasarımlarında hata değerlerini azaltmak için devre

elemanları tasarımlarda sınırsız değerler alabilmektedir fakat bu, devrelerin gerçeklenmesini zorlaştırmaktadır. Bundan dolayı, devre elemanları üretimde tipik olarak kullanılan serilerden (E12, E24, E48, E96) seçilmektedirler. Bu, tasarım maliyetini azaltırken devrenin kolaylıkla gerçeklenmesini olanaklı kılmaktadır. Aktif filtre devreleri çeşitli topolojilerde tasarlanabilmektedir. Literatürde en çok kullanılan topolojiler, durum değişgenli (SV), gerilim çoklu geri besleme (MFB) ve Sallen-Key aktif filtre topolojileridir.

Sayısal filtreler iki ana grupta incelenebilir: Sonlu darbe cevaplı (FIR) ve sonsuz darbe cevaplı (IIR) sayısal filtreler. FIR filtrelerde çıktı sadece girdilerin bir fonksiyonuyken (özyinelemesiz) IIR filtrelerde çıktı geçmişteki çıktıların bir fonksiyonu (özyinelemeli) olarak tanımlanmaktadır (Winder 2002). Şekil 1.1'de sayısal filtrelerin, $x(n)$ giriş işaretine karşılık $y(n)$ çıkış işaretinin özyinemeli ve özyinelemesiz yapısı gösterilmektedir.

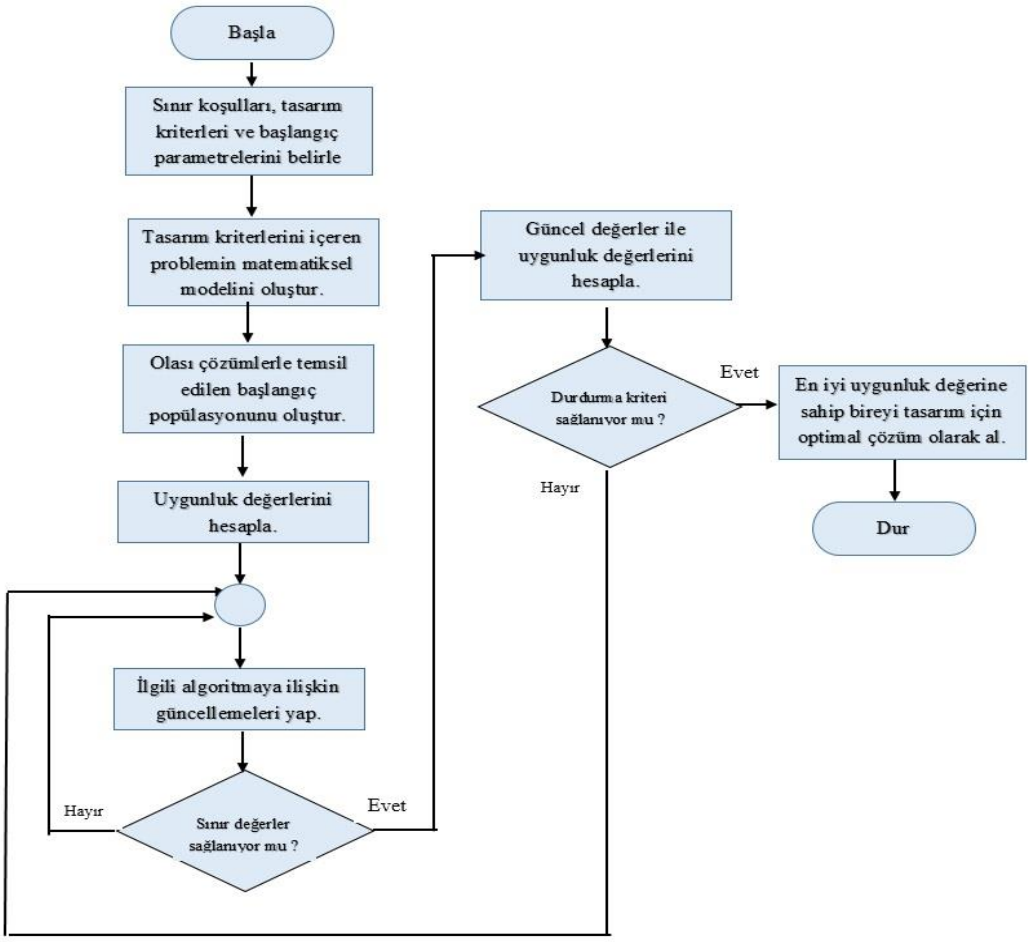


Şekil 1.1. Sayısal filtre yapıları

FIR filtreler tamamen kararlı yapıda olup doğrusal faz kayması göstermektedirler. Bu tür filtreler yalnızca geçmiş ve yürürlükteki girdileri kullanmakta ve analog dünyada karşılıkları bulunmamaktadır. Bu filtrelerin hata yüzeyleri transfer fonksiyonuna göre konveks yapıda olmaları nedeniyle (tek modlu) en iyi çözümü temsil eden tek bir nokta barındırmaktadırlar. IIR filtreleri daha düşük sayıda katsayılarla daha iyi performans üretmekte fakat FIR filtrelerinin daima kararlı olması, doğrusal faz gibi bazı avantajlarını daima yakalayamamaktadırlar. Diğer yandan, IIR filtrelerin hata yüzeylerinde filtre katsayılarına bağlı olarak küresel minimumu temsil eden çözümün yanı sıra yerel

minimumu temsil eden nokta veya noktalar bulunabilmektedirler (çok modlu). Bu nedenle IIR filtrelerin FIR filtrelere göre tasarımındaki katsayılarının doğrulukla hesaplanması daha zor olmaktadır (Haykin 2013). Sayısal filtre tasarımında temel amaç, filtrenin transfer fonksiyonuna bağlı olarak optimal pay ve payda katsayılarını hesaplamaktır (Schlichthatle 2000).

Evrimsel algoritmalar, klasik hesaplama yöntemlerinin yetersiz kaldığı veya farklı çözüm yollarının üretilmeye ihtiyaç duyulduğu noktalarda sıklıkla başvurulan yöntemlerin başında gelmektedir. Literatürde, evrimsel algoritmalar aracılığıyla analog ve sayısal filtre tasarımlarına ait birçok çalışma mevcuttur. Evrimsel algoritmaların bu çalışmalara ilişkin genel optimizasyon akış şeması Şekil 1.2'de verilmektedir.



Şekil 1.2. Filtre tasarım problemlerinde evrimsel algoritmaların genel optimizasyon akış şeması

Birçok alanda sıklıkla kullanılan genetik algoritma (Genetic algorithm-GA) (Holland 1975), diferansiyel gelişim algoritması (Differential evolution algorithm-DE) (Storn ve Price 1995) ve parçacık sürü optimizasyon algoritması (Particle swarm optimization algorithm-PSO) (Kennedy ve Eberhart 1995) analog filtre tasarımlarında da kullanılmaktadır. Bu algoritmalar ön tanımlı olarak belirlenen üretim serilerine göre analog filtre devre elemanlarını belirleyerek optimal analog filtre devresi tasarlamayı amaçlamaktadırlar (Horrocks ve Spittle 1993, Vural ve Yıldırım 2010, Vural ve ark. 2013). Kalınlı, SV aktif filtrelerin optimal devre elemanları seçimi için bağışıklık ve paralel tabu arama algoritmalarını önermiştir (Kalınlı 2004, 2006). Bu çalışmalarda, önerdiği algoritmaların SV aktif filtre devreleri üzerinde iki farklı tipte performansları analiz edilmiştir. Ele alınan ilk tipte tüm devre elemanları bağımsız olarak kabul edilmiş, diğer tipte ise bazı devre elemanları devreye harici olarak bağlanıp algoritmaların devre tasarımındaki performansları gözlemlenmiştir. Min ve arkadaşları klonal seçim algoritmasını (CSA) dördüncü dereceden gerilim kontrollü gerilim kaynağı (Voltage Controlled Voltage Source-VCVS) alçak geçiren Butterworth aktif filtre için önermişler ve bu filtre devresini E12 serisine göre tasarlamışlardır. (Min ve ark. 2007). Bu çalışmanın sonuçları, CSA algoritması tasarladıkları filtre devresinde GA, tabu arama algoritması ve geleneksel metotlardan daha iyi sonuçlar vermektedir. Analog filtre tasarımlarında, algoritmaların performanslarını kıyaslayan karşılaştırmalı çalışmalarda literatürde mevcuttur. Bu çalışmaların birinde (Vural ve ark. 2012), dördüncü dereceden VCVS alçak geçiren Butterworth aktif filtre yapay arı kolonisi (ABC), GA ve PSO algoritmalarıyla tasarlanmıştır. Bu çalışmada, PSO kıyaslanan algoritmalara nazaran daha düşük hatayla analog aktif filtre tasarımı yapmıştır. Fakat PSO'nun tasarım için bulduğu bazı devre elemanı değerleri üretim serileri dışında kalmaktadır. Varolan algoritmaların analog filtre tasarımlarında gösterdiği zayıflıklar, daha güçlü algoritmalara ihtiyacı doğurmuştur. Bu algoritmalar diğer algoritmalar gibi hatayı minimize ederek, analog filtre devresi için optimal devre elemanlarını bulmaya çalışmaktadırlar (Digbalay ve ark. 2014, Bishnu ve ark. 2015).

Evrimsel algoritmalar, sadece pay katsayılarından oluşmakta olan FIR filtre transfer fonksiyonuna bağlı olarak, en uygun filtre katsayıları seçerek filtreyi tasarlayabilmektedirler. Boudjelaba yaptığı çalışmada (Boudjelaba ve ark. 2014), FIR filtre tasarımlarında GA ve PSO algoritmalarını kıyaslamış ve hata kriterleri açısından PSO algoritmasının genellikle GA'dan daha iyi olduğunu gözlemlemiştir. Diğer bir evrimsel algoritma olan tavlama benzetimi algoritmasıyla FIR filtre tasarımları yapılmış ve bu algoritmanın hata yüzeylerinde global minimum noktasına iyi bir yakınsama gösterdiği görülmüştür (Kacelenga ve ark. 1990). Karaboğa ve Çetinkaya (2006) yaptığı çalışmada, DE algoritması ile gerçeklediği sekiz, on dört ve yirminci dereceden FIR filtre tasarımlarında, bu algoritmanın GA ve klasik algoritmalara nazaran daha az hatayla filtre tasarımı yapabildiğini göstermiştir. Daha sonraki yıllarda, Lui ve ark. (2010) DE algoritmasını geliştirerek FIR filtre tasarımları için kullanmışlardır. Bu çalışma, geliştirdikleri algoritmanın orijinal DE algoritmasına göre daha iyi bir yakınsama hızı ve daha iyi bir performans elde edebildiğini göstermişlerdir.

IIR filtrelerin kuadratik olmayan çok modlu hata yüzeylerine sahip olması, bazı durumlarda klasik hesaplama yöntemlerinin yetersiz kalmasına ve evrimsel algoritmalara ihtiyaç duyulması sonucunu doğurmuştur. Literatürde sıklıkla kullanılan GA, DE ve PSO algoritmaları IIR filtre tasarımlarında ve adaptif IIR filtre vasıtasıyla sistem modellemede başarıyla kullanılmışlardır (Karaboga 2005, Tsai ve ark. 2009, Chen ve Luk 2010). Bu algoritmaların, IIR hata yüzeyinin sıklıkla birden çok yerel minimum içermesi sebebiyle hibrit versiyonları oluşturulmuş ve tasarımlarda daha iyi sonuçlar elde edilmesi amaçlanmıştır. Bu amaçla, Singh ve ark. (2013) yaptıkları çalışmada, IIR filtre tasarımları için DE algoritmasını model arama tekniğiyle hibritleştirmiş ve daha az hataya sahip tasarımlar elde etmiştir. Diğer bir DE algoritmasının hibritleştirildiği çalışmada (Kaur ve Dhillon 2014), bu algoritma kedi sürüsü optimizasyon algoritmasıyla hibritleştirilmiş ve hibrit algoritmanın IIR filtre tasarımlarında orijinallerine nazaran daha iyi performans gösterdiği gözlemlenmiştir. IIR filtre tasarımları için yeni algoritmalarda geliştirilmiştir. Yu ve Xinjie (2007) yaptıkları çalışmada, IIR filtre tasarımla için yeni bir genetik algoritma varyantı önermişlerdir. Bu çalışmada önerdikleri algoritma, filtrenin genlik ve

faz cevaplarını eş zamanlı optimize ederek başarılı sonuçlara ulaşmaktadır. Saha ve arkadaşları dalgacık mutasyonu kullanarak yerçekimi arama algoritması ile IIR filtre tasarımlarında karşılaştırdıkları diğer yöntemlere göre daha iyi bir performans elde ettiklerini göstermişlerdir (Saha ve ark. 2015).

Yapılan bu tez çalışmasında, literatürde sıklıkla kullanılan GA (Holland 1975), DE (Storn ve Price 1995), PSO (Kennedy ve Eberhart 1995), karınca koloni optimizasyon algoritması (Ant colony optimization algorithm-ACO) (Dorigo 1992), armoni arama algoritması (Harmony search algorithm-HS) (Geem ve ark. 2001), girdap arama algoritması (Vortex search algorithm-VS) (Doğan ve Ölmez 2015) ve geri izleme arama optimizasyon algoritması (Backtracking search algorithm-BS) (Civicioglu 2013) ile farklı derecelerden FIR ve IIR sayısal filtre, farklı derece ve topolojilerde analog filtre tasarımları gerçekleştirilmiştir. Sayısal filtre tasarımlarının uygunluğu MATLAB programı aracılığıyla ilişkili algoritmanın frekans cevabı grafiğiyle doğrulanmıştır. Analog filtre tasarımlarında ise PSpice programı aracılığıyla algoritmaların bulduğu, ilgili üretim serisine ait devre elemanları değerlerine göre filtre devreleri gerçeklenmiş ve programın benzetim grafikleriyle tasarlanan devrelerin doğrulukları gösterilmiştir.

Tez çalışmasının ikinci bölümünde, genel filtre tipleri, filtrelerde kullanılan terimler, analog aktif filtreler hakkında genel bilgiler, aktif filtre devrelerini gerçeklemede literatürde sıklıkla kullanılan topolojilerden bahsedilecek, sayısal filtre yapıları detaylı olarak anlatılacak, sayısal filtre tasarım yaklaşımlarından söz edilecektir. Üçüncü bölümde, filtre tasarımlarında kullanılan evrimsel algoritmaların çözümlerini elde etmede kullandığı yöntemler ve algoritmaların işlem adımları açıklanacaktır. Dördüncü bölümde, analog ve sayısal filtre tasarımlarına ilişkin tasarım örneklerine yer verilecek ve bu örnekler üzerinde algoritmaların performanslarının detaylı analizleri yapılacaktır. Beşinci bölümde sonuçlar kısmına yer verilecektir.

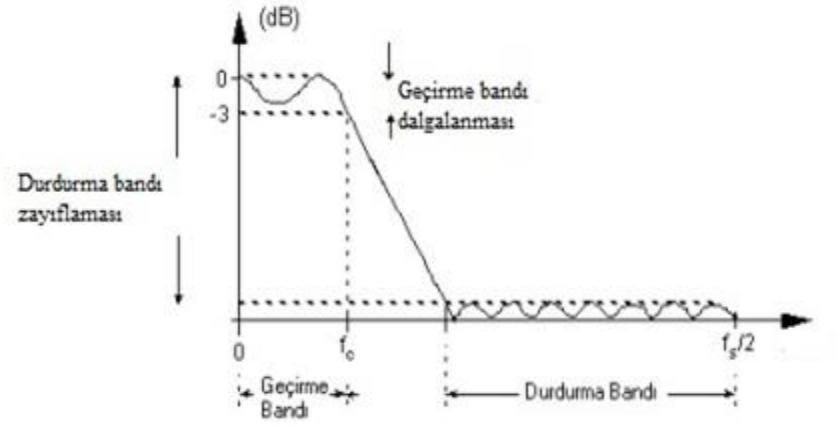
2. FİLTRELER

2.1. Giriş

Filtreler bazı işaretlerin geçmesini ve bazı işaretlerin durdurulmasını sağlamaktadır. Girişe uygulanan işaret frekanslarının çok az azaltma ile ya da hiç azaltma olmaksızın çıkış üzerinden geçmesine imkan tanımaktadır. Filtrelerde kullanılan bazı terimler aşağıda özetlenmektedir.

- Geçirme bandı (passband): Filtre girişine uygulanan işaretin tam olarak geçirildiği frekans bölgesine verilen isimdir.
- Geçirme bandı dalgalanması (passband ripple): Filtrenin geçirme bandında oluşan istenmeyen dalgalanmalar bütünüdür.
- Durdurma bandı (stopband): Filtre girişine uygulanan işaretin ilerletilmeyip durdurulduğu frekans bölgesine verilen isimdir.
- Durdurma bandı zayıflaması (stopband attenuation): Filtre girişine uygulanan işaretin genişleme zayıflamasıdır.
- Kesim frekansı (cutoff frequency): Gerilim kazancının 3 dB zayıfladığı nokta veya noktalar olarak tanımlanmaktadır.

Şekil 2.1'de yukarıdaki terimlere ilişkin gösterimlerle alçak geçiren bir filtrenin frekans cevabı verilmektedir.



Şekil 2.1. Alçak geçiren filtrenin frekans cevabı (Batık 2011)

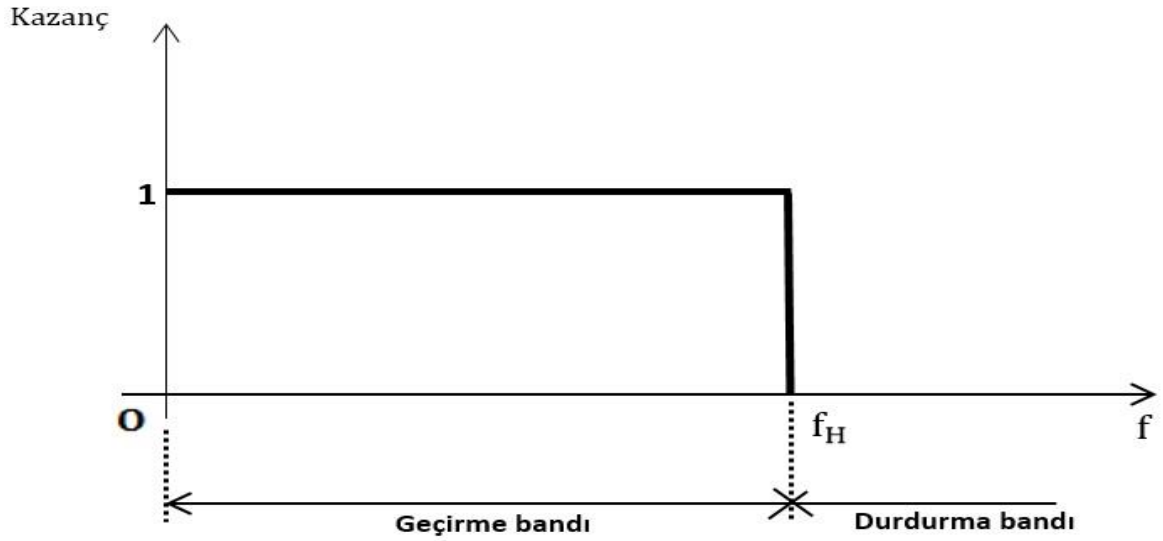
Şekil 2.1'den görüldüğü üzere, alçak filtrenin kazancının -3 dB'ye düştüğü nokta kesim frekansı olarak adlandırılmaktadır.

2.1.1 Alçak geçiren filtre

Alçak geçiren filtreler belirli bir frekans değerinin altındaki işareti geçiren ve bu frekans değerinin üstündeki işaretleri geçirmeyen filtrelerdir. Yüksek frekans bileşenlerinin süzülmesi gereken durumlarda bu tip filtre kullanılmaktadır (Lacanette 2010). İdeal alçak geçiren filtre matematiksel olarak Denklem (2.1)'deki gibi tanımlanabilir.

$$G = \begin{cases} 0 & f \rangle f_h \quad Durdurma\ bandı \\ 1 & 0 \leq f \leq f_h \quad Geçirme\ bandı \end{cases} \tag{2.1}$$

Burada, G filtrenin kazancını, f_h filtrenin kesim frekansını belirtmektedir. Şekil 2.2'de ideal alçak geçiren filtrenin frekans karakteristik grafiği verilmektedir. Grafiktende görülebileceği üzere kesim frekansı içerisindeki frekanslar geçirme bandı olarak tanımlanırken, bu frekansın üzerindeki frekanslar durdurma (söndürme) bandı olarak adlandırılmaktadır.



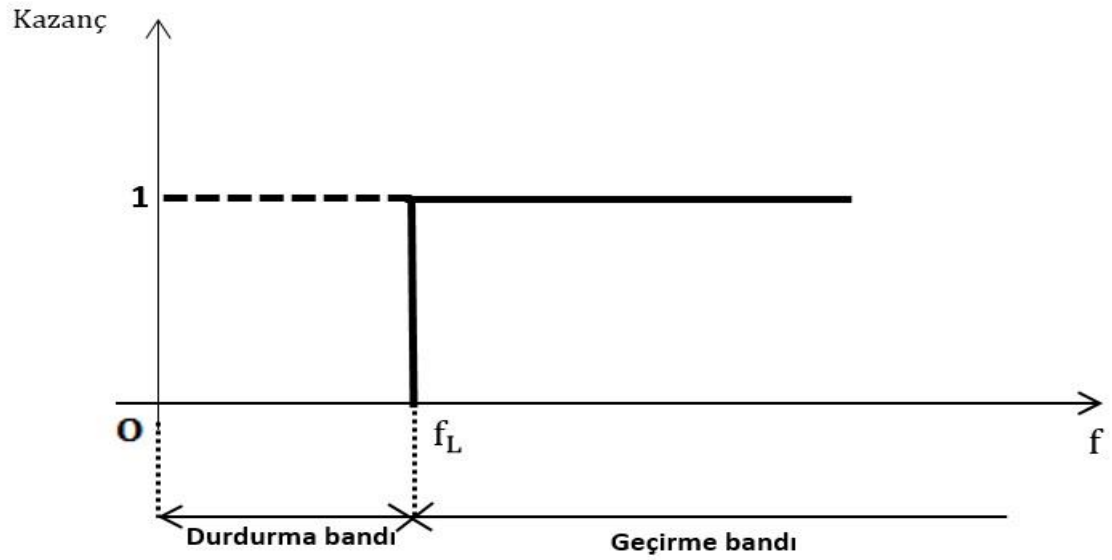
Şekil 2.2. İdeal alçak geçiren filtre

2.1.2 Yüksek geçiren filtre

Yüksek geçiren filtreler belirli bir frekansın üstündeki işaretleri geçiren ve bu frekansın altındaki işaretleri geçirmeyen filtrelerdir. İşaretlerin düşük frekans bileşenlerinin süzülmesi gereken uygulamalarda kullanılmaktadır (Lacanette 2010). İdeal yüksek geçiren filtre matematiksel olarak Denklem (2.2)'deki gibi tanımlanabilir.

$$G = \begin{cases} 0 & 0 \langle f \langle f_L \quad Durdurma\ bandı \\ 1 & f \geq f_L \quad Geçirme\ bandı \end{cases} \tag{2.2}$$

Denklem (2.2)'de, G filtrenin kazancını, f_L filtrenin kesim frekansını belirtmektedir. Şekil 2.3'te ideal yüksek geçiren filtrenin frekans karakteristik grafiği verilmektedir. Grafikten görülebileceği üzere kesim frekansı altındaki frekanslarda işaret sönümlendirilirken, bu frekansın üzerindeki frekanslarda işaretin geçişine izin verilmektedir.



Şekil 2.3. İdeal yüksek geçiren filtre

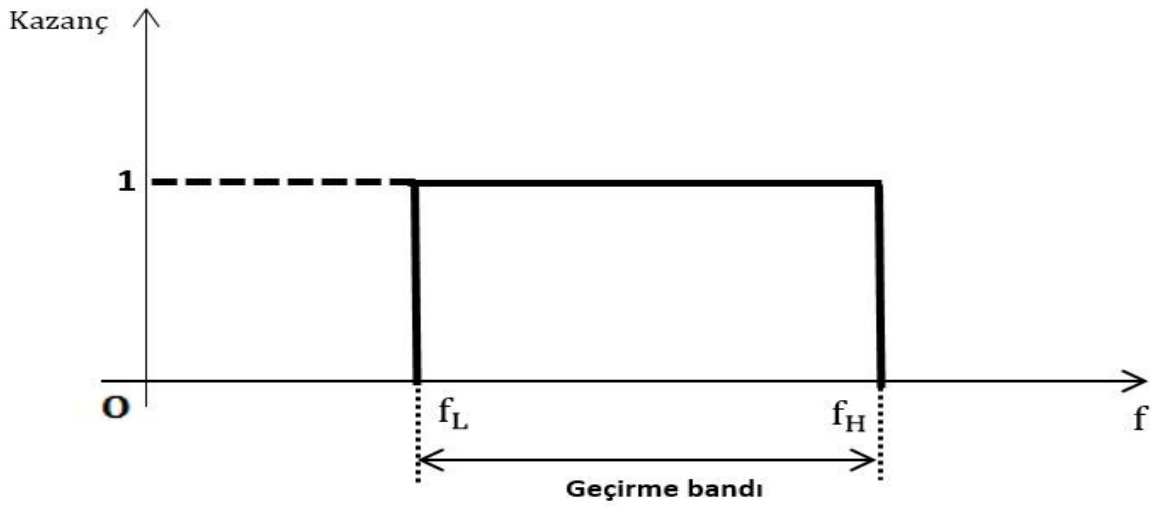
2.1.3 Band geçiren filtre

Bant geçiren filtre belirlenen frekans aralığındaki frekanslı işaretleri geçiren, bu frekans aralığı dışındaki işaretleri geçirmeyen filtrelerdir. Belirli bir frekanstaki işareti ayırmak veya bir frekans bandındaki işaretleri diğer frekanslardaki işaretlerden ayrıştırmak için

kullanılabilmektedirler (Lacanette 2010). İdeal band geçiren filtre matematiksel olarak Denklem (2.3)'teki gibi tanımlanabilir.

$$G = \begin{cases} 0 & Diğer\ durumda \quad Durdurma\ bandı \\ 1 & f_L \leq f \leq f_H \quad Geçirme\ bandı \end{cases} \tag{2.3}$$

Yukarıdaki denklemde, G filtrenin kazancını, f_L ve f_H filtrenin alt ve üst kesim frekanslarını belirtmektedir. Şekil 2.4'te ideal band geçiren filtrenin frekans karakteristik grafiği verilmektedir. Bu filtre tipinde, alt ve üst kesim frekansları arasındaki frekanslar geçirme bandı olarak, bu frekansın dışında kalan frekanslar ise durdurma bandı olarak adlandırılır.



Şekil 2.4. İdeal bant geçiren filtre

2.1.4 Band durduran filtre

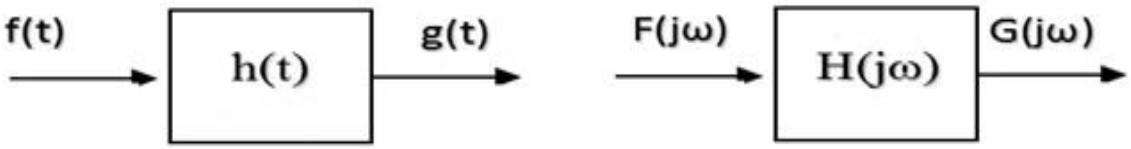
Bant durduran filtre, belirlenen frekans aralığı arasındaki işaretleri geçirmeyen, bu frekans aralığının altındaki ve üstündeki frekanslardaki işaretleri geçiren filtrelerdir. İşaretin diğer frekanslardaki bileşenleri mümkün olduğunca az seviyede etkileyecek şekilde, bir işaretdeki istenmeyen frekansları süzmek için kullanılabilmektedir (Lacanette 2010). İdeal band durduran filtre Denklem (2.4)'teki gibi tanımlanabilir.

$$G = \begin{cases} 1 & Diğer\ durumda \\ 0 & f_L \langle f \langle f_H \end{cases} \quad \begin{matrix} Durdurma\ bandı \\ Geçirme\ bandı \end{matrix} \tag{2.4}$$

Denklem (2.4)'te, G filtrenin kazancını, f_L ve f_H filtrenin alt ve üst kesim frekanslarını belirtmektedir. Band durduran filtrenin alt ve üst kesim frekansları arasında işaret sönümlendirilirken, bu frekans aralığı dışındaki frekanslarda geçişine izin verilmektedir.

2.2. Analog Filtreler

Tek girişli tek çıkışlı sürekli zamanlı sistem Şekil 2.5'teki gibi gösterilmektedir.



Şekil 2.5. Tek girişli tek çıkışlı sürekli zamanlı sistem

Sistemin giriş ve çıkış işaretlerine Fourier dönüşümü uygulanırsa Denklem (2.7) elde edilmektedir.

$$f(t) \longleftrightarrow F(j\omega) \tag{2.5}$$

$$g(t) \longleftrightarrow G(j\omega) \tag{2.6}$$

Denklem (2.5) ve (2.6) eşitliklerinden sistemin transfer fonksiyonu;

$$H(j\omega) = \frac{G(j\omega)}{F(j\omega)}$$

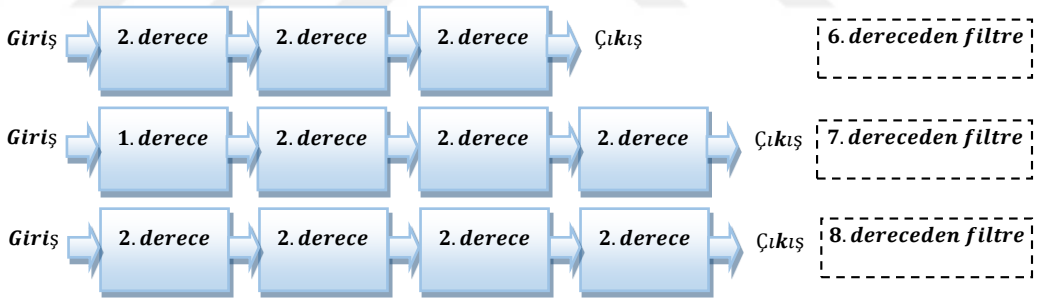
$$= |H(j\omega)|\exp(j\varphi(\omega)) \tag{2.7}$$

olarak elde edilmektedir. $|H(j\omega)|$ sistemin genlik cevabı ve $\varphi(\omega)$ sistemin faz cevabını belirtmektedir (Baher 2012). Analog filtre devreleri, belirli bir frekansın üzerindeki ya da altındaki frekansları veya bir frekans bandını tamamen durdurmak için kullanılan, temel olarak aktif ve pasif olarak iki gruba ayrılan filtre devreleridir. Aktif analog filtre devreleri

genellikle işlemsel yükseltgeç (opamp), direnç ve kondansatör gruplarının uygun şekilde bağlanmasıyla oluşurken, pasif filtreler direnç, bobin ve kondansatörlerden oluşmaktadır. Analog aktif filtre devreleri çeşitli topolojilerde tasarlanabilmektedir (Mancini 2002). Bu bölümde literatürde sıklıkla kullanılan Sallen-Key, MFB ve SV topolojileri ile oluşturulan aktif filtre devreleri ele alınacaktır.

2.2.1. Aktif analog filtreler

Aktif filtreleri "aktif" yapan devre elemanı yapısında bulundurduğu işlemsel kuvvetlendiricilerdir. İşlemsel kuvvetlendiricilerin düşük çıkış empedansı ve yüksek giriş empedansına sahip olmasından dolayı sistem elemanları arasındaki etkileşimi azaltarak, etkili bir izolasyon sağlamaktadırlar. Yüksek dereceden analog filtreler düşük dereceden filtrelerin ardışık bağlanmasıyla elde edilebilmektedir (Mancini 2002). Şekil 2.6'da aktif analog filtrelerin yüksek dereceden filtreleri elde etmede ardışık bağlanmasına ilişkin blok diyagramı verilmiştir.

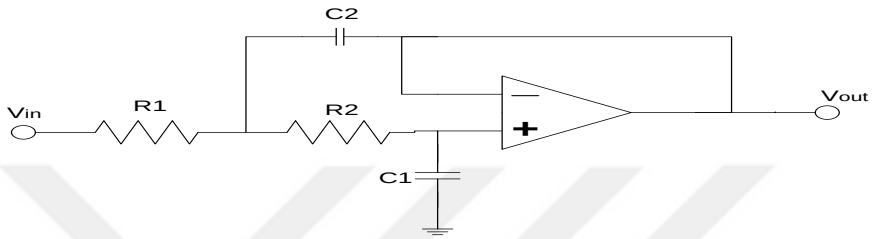


Şekil 2.6. Yüksek dereceden analog filtrelerin ardışık yapı blokları

Şekil 3.2'deki blok diyagramından görüldüğü üzere, yüksek dereceli aktif analog filtreler birinci ve ikinci dereceli düşük dereceli eşdeğer devreleri kullanılarak tasarlanabilmektedirler. Şekilde, altı, yedi ve sekizinci dereceden filtrelerin, bir ve ikinci dereceden filtrelerin birbirlerine ardışık bağlanmasıyla oluşturulabileceği gösterilmektedir. Aktif analog filtreler farklı topolojiler kullanarak tasarlanabilmektedir. Bu bölümde ilgili topolojilerden Sallen-Key, SV ve MFB topolojileri incelenecektir.

2.2.1.1. Alçak geçiren aktif analog filtre devreleri

Alçak geçiren aktif analog filtre Sallen-Key filtre topolojisini kullanarak tasarlanabilmektedir. Şekil 2.7'de ikinci dereceden alçak geçiren Sallen-Key filtresi verilmektedir. Filtre devresinin girişi V_{in}, filtre çıkışı V_{out} ile gösterilmektedir.



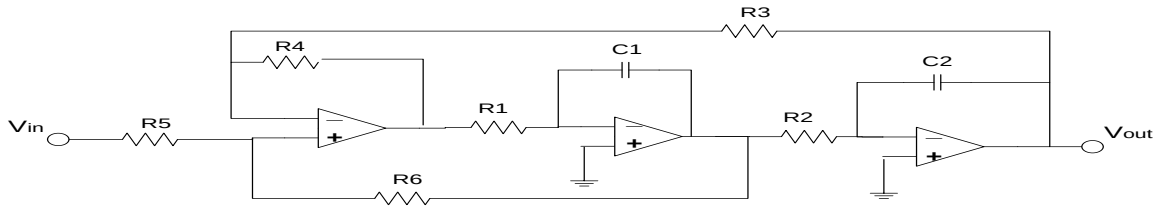
Şekil 2.7. İkinci dereceden Sallen-Key alçak geçiren filtre devresi

Şekil 2.7'deki filtre devresinin transfer fonksiyonu Denklem (2.8)'de verilmektedir (Mancini 2002).

$$A(s) = \frac{1}{1+w_c C_1(R_1+R_2)s+w_c^2 C_1 C_2 R_1 R_2 s^2} \tag{2.8}$$

Denklem (2.8)'de, w_c alçak geçiren filtrenin kesim frekansını ifade etmektedir.

SV topolojisini kullanarak aktif analog filtre tasarlanabilmektedir. Şekil 2.8'de SV topolojiyle tasarlanmış ikinci dereceden alçak geçiren filtre verilmektedir.

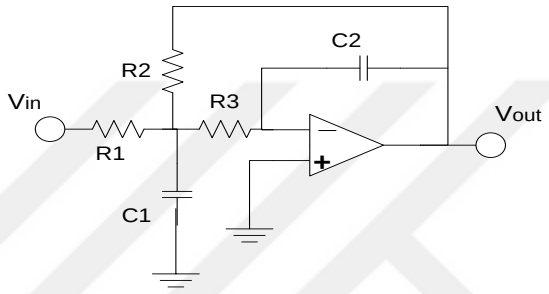


Şekil 2.8. İkinci dereceden durum değişkenli alçak geçiren filtre devresi

İkinci dereceden SV alçak geçiren filtre devresinin transfer fonksiyonu Denklem (2.9)'da verilmektedir (Anonim 2008).

$$A(s) = \frac{\left(\frac{1+\frac{R_3}{R_4}}{1+\frac{R_5}{R_6}}\right)\left(\frac{R_4}{R_3}\right)\left(\frac{1}{R_1R_2C_1C_2}\right)}{s^2+\left(\frac{1+\frac{R_4}{R_3}}{1+\frac{R_6}{R_5}}\right)\left(\frac{1}{R_1C_1}\right)s+\left(\frac{R_4}{R_3}\right)\left(\frac{1}{R_1R_2C_1C_2}\right)} \quad (2.9)$$

MFB topolojisi ile aktif analog filtre tasarlanabilmektedir. Şekil 2.9'da bu topolojiyle tasarlanmış ikinci dereceden alçak geçiren filtre verilmektedir.



Şekil 2.9. İkinci dereceden gerilim çoklu geri besleme alçak geçiren filtre devresi

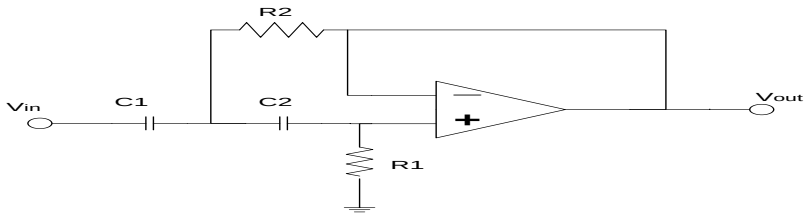
Denklem (2.10)'da Şekil 2.9'daki devrenin transfer fonksiyonu verilmektedir (Hercules 2012).

$$A(s) = \frac{\frac{1}{R_1R_3C_1C_2}}{s^2+\frac{1}{C_1}\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}\right)s+\frac{1}{R_2R_3C_1C_2}} \quad (2.10)$$

Filtre devresinin girişi V_{in}, çıkışı V_{out} ile ifade edilmektedir.

2.2.1.2. Yüksek geçiren aktif analog filtre devreleri

Yüksek geçiren aktif analog filtre Sallen-Key filtre topolojisini kullanarak tasarlanabilmektedir. Şekil 2.10'da ikinci dereceden alçak geçiren Sallen-Key filtresi verilmektedir.

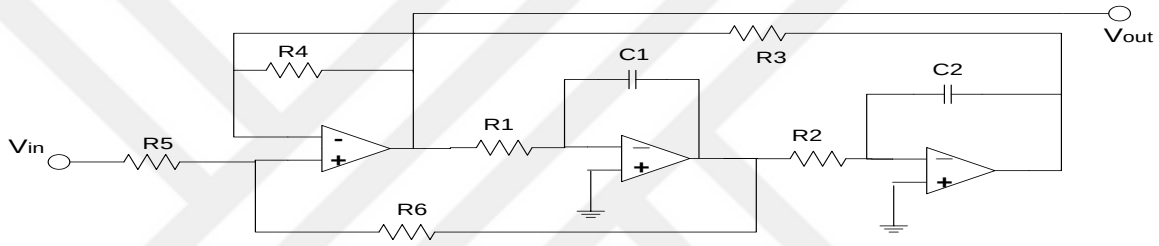


Şekil 2.10. İkinci dereceden Sallen-Key yüksek geçiren filtre devresi

Şekil 2.10'daki filtre devresinin transfer fonksiyonu Denklem (2.11)'de verilmektedir (Mancini 2002).

$$A(s) = \frac{1}{1+\frac{R_2(C_1+C_2)}{w_c C_1 C_2 R_1 R_2 s}+\frac{1}{w_c^2 C_1 C_2 R_1 R_2 s^2}} \tag{2.11}$$

Denklem (2.11)'de, w_c yüksek geçiren filtrenin kesim frekansını ifade etmektedir. SV topolojisini kullanarak aktif analog filtre tasarlanabilmektedir. Şekil 2.11'de bu topolojiyle tasarlanmış ikinci dereceden yüksek geçiren filtre verilmektedir.

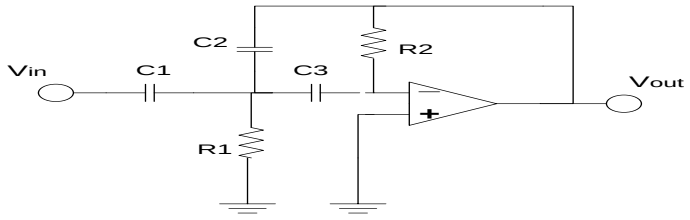


Şekil 2.11. İkinci dereceden durum değişkenli yüksek geçiren filtre devresi

İkinci dereceden SV yüksek geçiren filtre devresinin transfer fonksiyonu Denklem (2.12)'de verilmektedir (Anonim 2008).

$$A(s) = \frac{\left(\frac{1+\frac{R_4}{R_3}}{1+\frac{R_5}{R_6}}\right)s^2}{s^2+\left(\frac{1+\frac{R_4}{R_3}}{1+\frac{R_6}{R_5}}\right)\left(\frac{1}{R_1C_1}\right)s+\left(\frac{R_4}{R_3}\right)\left(\frac{1}{R_1R_2C_1C_2}\right)} \tag{2.12}$$

MFB topolojisi ile aktif analog filtre tasarlanabilmektedir. Şekil 2.12'de bu topolojiyle tasarlanmış ikinci dereceden yüksek geçiren filtre verilmektedir.



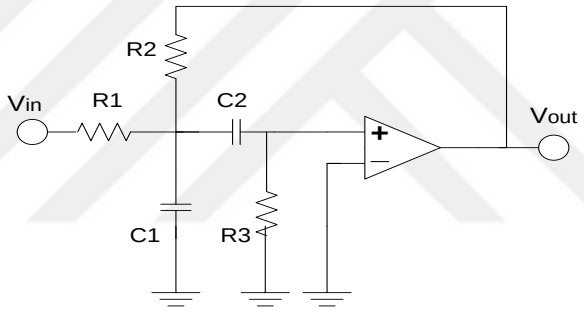
Şekil 2.12. İkinci dereceden gerilim çoklu geri besleme yüksek geçiren filtre devresi

Denklem (2.13)'te, Şekil 2.12'deki devrenin transfer fonksiyonu verilmektedir (Hercules 2012).

$$A(s) = \frac{-\frac{C_1}{C_2}s^2}{s^2 + \frac{1}{R_2}\left(\frac{C_1}{C_2C_3} + \frac{1}{C_2} + \frac{1}{C_3}\right)s + \frac{1}{R_1R_2C_2C_3}} \tag{2.13}$$

2.2.1.3. Band geçiren aktif analog filtre devreleri

Band geçiren aktif analog filtre Sallen-Key filtre topolojisini kullanarak tasarlanabilmektedir. Şekil 2.13'te ikinci dereceden alçak geçiren Sallen-Key filtresi verilmektedir.

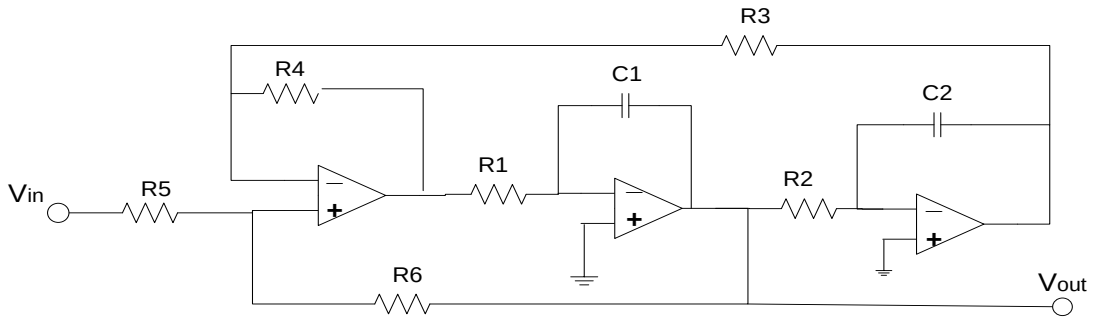


Şekil 2.13 İkinci dereceden Sallen-Key band geçiren filtre devresi

Şekil 2.13'teki filtre devresinin transfer fonksiyonu Denklem (2.14)'te verilmektedir (Hercules 2012).

$$A(s) = \frac{\frac{1}{R_1C_1}s}{s^2 + \left(\frac{1}{R_1C_1} + \frac{1}{R_3C_2} + \frac{1}{R_3C_1}\right)s + \frac{R_1+R_2}{R_1R_2C_1C_2}} \tag{2.14}$$

SV topolojisini kullanarak aktif analog filtre tasarlanabilmektedir. Şekil 2.14'te SV topolojiyle tasarlanmış ikinci dereceden band geçiren filtre verilmektedir.

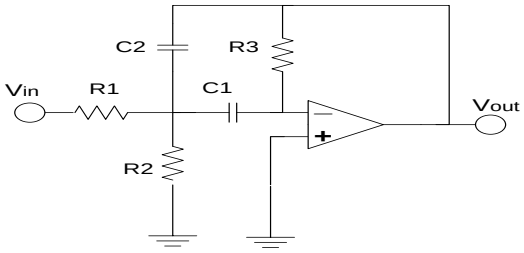


Şekil 2.14. İkinci dereceden durum değişkenli band geçiren filtre devresi

İkinci dereceden SV band geçiren filtre devresinin transfer fonksiyonu Denklem (2.15)'te verilmektedir (Anonim 2008).

$$A(s) = \frac{-\left(\frac{R_6}{R_5}\right)\left(\frac{1+\frac{R_4}{R_3}}{1+\frac{R_6}{R_5}}\right)\left(\frac{1}{R_1C_1}\right)s}{s^2+\left(\frac{1+\frac{R_4}{R_3}}{1+\frac{R_6}{R_5}}\right)\left(\frac{1}{R_1C_1}\right)s+\left(\frac{R_4}{R_3}\right)\left(\frac{1}{R_1R_2C_1C_2}\right)} \tag{2.15}$$

MFB topolojisi ile aktif analog filtre tasarlanabilmektedir. Şekil 2.15'te MFB topolojiyle tasarlanmış ikinci dereceden band geçiren filtre verilmektedir.



Şekil 2.15. İkinci dereceden gerilim çoklu geri besleme band geçiren filtre devresi

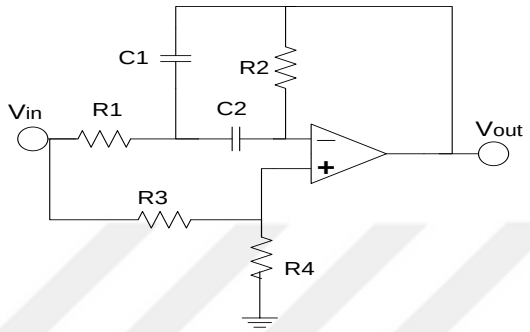
Denklem (2.16)'da Şekil 2.15'teki devrenin transfer fonksiyonu verilmektedir (Hercules 2012).

$$A(s) = \frac{-\frac{1}{R_1C_2}s}{s^2+\frac{1}{R_3}\left(\frac{1}{C_1}+\frac{1}{C_2}\right)s+\frac{1}{R_3C_1C_2}\left(\frac{1}{R_1}+\frac{1}{R_2}\right)} \tag{2.16}$$

Filtre devresinin girişi V_{in}, çıkışı V_{out} ile ifade edilmektedir.

2.2.1.4 Band durduran aktif analog filtre devreleri

Band durduran aktif analog filtre MFB filtre topolojisini kullanarak tasarlanabilmektedir. Şekil 2.16'da ikinci dereceden band durduran MFB filtre devresi verilmektedir.



Şekil 2.16. İkinci dereceden gerilim çoklu geri besleme band durduran filtre devresi

Şekil 2.16'daki filtre devresinin transfer fonksiyonu Denklem (2.17)'de verilmektedir (Hercules 2012).

$$A(s) = \frac{R_4}{R_3+R_4} \frac{s^2+\left(\frac{1}{R_2C_1}+\frac{1}{R_2C_2}-\frac{R_3}{R_4R_1C_1}\right)s+\frac{1}{R_1R_2C_1C_2}}{s^2+\left(\frac{1}{R_2C_1}+\frac{1}{R_2C_2}\right)s+\frac{1}{R_1R_2C_1C_2}} \quad (2.17)$$

Filtre devresinin girişi V_{in}, çıkışı V_{out} ile ifade edilmektedir.

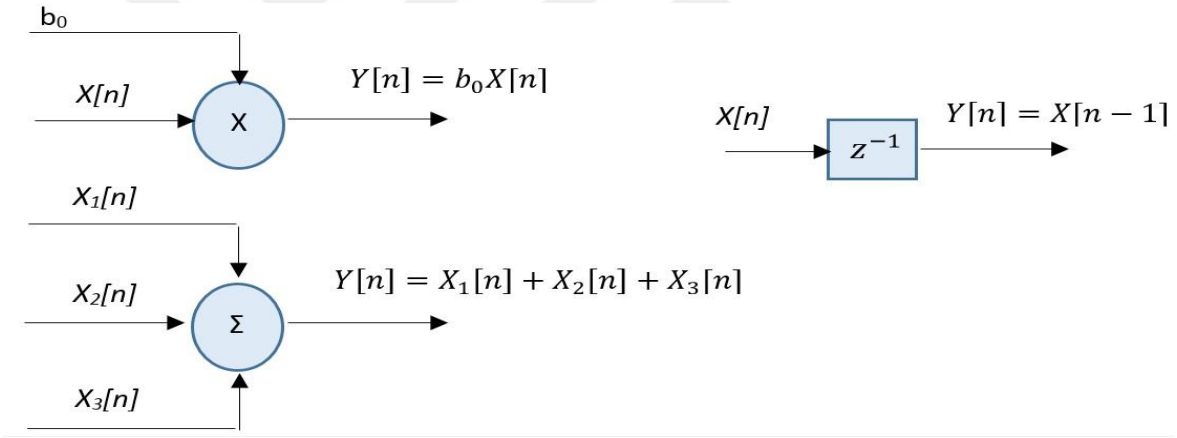
2.3. Sayısal Filtreler

Sayısal filtreleme, sayısal işaret işlemede yaygın olarak kullanılan bir işlem olup uygulaması nispeten kolaydır. Sayısal filtreler ayrık zamanlı işaretlerin ilgili frekans spektrumu üzerinde istenilen işlemleri yapan (düşük gürültü, distorsiyon vb.) ve işareti daha istenen özelliklere sahip işaret biçimine getiren yapılardır. Sayısal filtrelerin kararlı olması ve işaretin bozulmaya uğramadan filtrelemeden geçebilmesi için filtre transfer fonksiyonunun faz cevabının frekansa göre doğrusal olması beklenmektedir. Sayısal filtreler birim dürtü cevaplarına göre sonlu dürtü cevaplı (Finite Impulse Response-FIR) ve sonsuz dürtü cevaplı (Infinite Impulse Response-IIR) olmak üzere ikiye ayrılmaktadırlar. Ayrıca, sayısal filtreler sahip oldukları doğrusal, doğrusal olmayan, zamanla değişen,

zamanla değişmeyen gibi diğer özelliklere göre de sınıflandırılabilmektedirler (Ertürk 2009). Sayısal filtreler, üç basit elemanın birbirleriyle bağlantısından oluşmaktadır:

- Çarpıcılar, bilgisayarın aritmetik mantık biriminde de kullanılan bir operatör olup girişine gelen girdiyi genellikle filtre katsayısı olarak bilinen bir katsayı ile çarpma işlemini yürütmektedir.
- Toplayıcılar, çarpıcı çıkışlarını toplayarak birleştirici rol üstlenmektedirler. Toplam filtre çıkışını oluşturmaktadırlar.
- Geciktiriciler, z^{-1} gecikme operatörüyle belirtilmektedirler. Sıralamadaki geçmiş ve gelecekteki değerlerin erişimine izin veren bileşenlerdir.

Şekil 2.17'de bu elemanların işlemsel blok diyagramı verilmektedir.



Şekil 2.17. Sayısal filtreleme işlemlerinde kullanılan operatörler

Sayısal filtre girişi $x(n)$, çıkışı $y(n)$ olmak üzere sabit katsayılı fark denklemi Denklem (2.18)'de verilmektedir (Ertürk 2009).

$$y[n] = \sum_{k=0}^{M} a_k x[n-k] - \sum_{k=1}^{N} b_k y[n-k] \tag{2.18}$$

Denklem (2.18)'e z-dönüşümü uygulanırsa,

$$Y(z) = H(z)X(z) \tag{2.19}$$

denklemi elde edilir. Buradan sayısal filtrenin transfer fonksiyonu;

$$H(z) = \frac{\sum_{k=0}^{M} a_k z^{-k}}{1+\sum_{k=1}^{N} b_k z^{-k}} \quad (2.20)$$

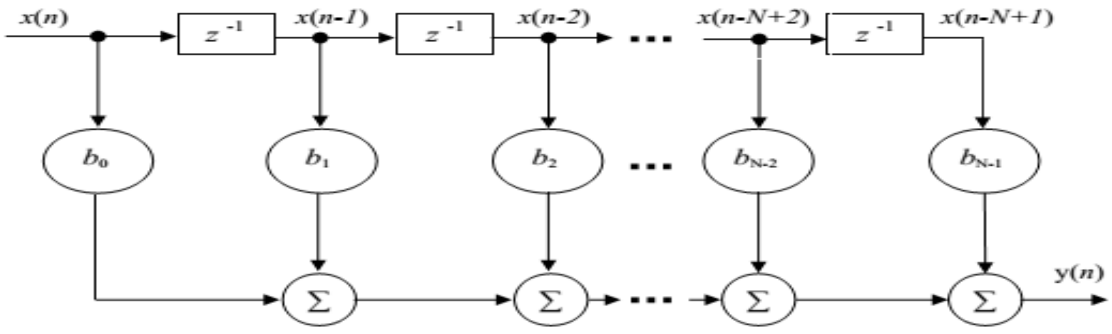
elde edilmektedir. Sayısal filtrenin frekans cevabı Denklem (2.20)'de z değişkeni yerine $z = e^{jw}$ koyularak Denklem (2.21)'deki gibi elde edilir.

$$H(e^{jw}) = \frac{\sum_{k=0}^{M} a_k e^{-jwk}}{1+\sum_{k=1}^{N} b_k e^{-jwk}} \quad (2.21)$$

$H(e^{jw})$ sayısal filtrenin frekans domeni karakteristiğini temsil eden frekans cevabını, a_k ve b_k filtre katsayılarını temsil etmektedir. Sayısal filtrenin frekans cevabı, $H(e^{jw})$ 'nin periyodu 2π olup, ters Fourier dönüşümü alındığında filtrenin dürtü cevabına ($h(\mathrm{n})$) ulaşılmaktadır. Sayısal filtrenin zaman domeni karakteristliği bu dürtü cevabı ile temsil edilmektedir.

2.3.1. FIR filtreler

Sonlu dürtü cevabına sahip FIR filtrelerde çıkış işareti sadece giriş işaretinin bir fonksiyonudur. Çıkış işareti, FIR filtreye gelen giriş işaretinin, z^{-1} gecikme operatörüyle tanımladığımız gecikme öğelerinden geçip daha sonra genellikle filtre katsayıları olarak ifade edilen bir sayı ile çarpılıp, giriş işaretinin bir fonksiyonu olarak oluşturulmaktadır. FIR filtreler, giriş işareti her bir gecikme operatöründen geçtiği için doğrusal fazlı olarak adlandırılırlar. Bundan dolayı işaretin tüm frekans bileşenleri aynı düzeyde geciktirilir yani grup gecikmesi sabittir. Şekil 2.18'de FIR filtre mimarisi verilmektedir.



Şekil 2.18. FIR filtre mimarisi (Çetinkaya 2010)

Şekil 2.18'den görüleceği üzere, FIR filtrenin iç yapısında Şekil 2.17'de verilen operatörler kullanılmaktadır. Burada eğer k örnek periyoduyla geciktirilen giriş işareti $x[n-k]$ ile temsil edilirse filtrenin çıkışı;

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots . + b_n x[0] \tag{2.22}$$

şeklinde elde edilir. Giriş işaretinin sıklıkla gecikme operatörlerini kullanarak uygun formlarının elde edilip, her bir formu filtre katsayıları ile çarparak, çıkan işaretlerin toplamından FIR filtrenin çıkışı elde edilmektedir. Bu işlem matematiksel ifadeyle konvolüsyon anlamına gelir. Filtrenin girişine uygulanan işaretin, filtre katsayılarını temsil eden $b_0, b_1, \ldots b_k$ değerleri ile konvolüsyon işlemine tabi tutulmasıdır. Denklem (2.23)'te konvolüsyon işlemin genel matematiksel eşitliği verilmektedir.

$$y[n] = \sum_{k=-\infty}^{\infty} h[k] x[n-k] \tag{2.23}$$

Bu işlem kullanılarak FIR filtrenin giriş çıkış ilişkisi Denklem (2.24)'teki gibi verilebilir (Winder 2002).

$$y[n] = \sum_{k=0}^{N-1} b_k x[n-k] \tag{2.24}$$

$x[n]$ giriş, $y[n]$ çıkış işaretini belirtmektedir. $N-1$ olarak gösterilmiş olan gecikme operatörleri sayısı filtrenin derecesini belirlemektedir. N ise filtrenin uzunluğu olup filtre katsayıları sayısına eşittir.

Eğer Denklem (2.24)'e z-dönüşümü uygulanırsa,

$$\begin{aligned} H(z) &= \sum_{k=0}^{N-1} b_k z^{-k} \\ &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{N-1} z^{-(N-1)} \end{aligned} \tag{2.25}$$

elde edilmektedir. Fitrenin frekans cevabı,

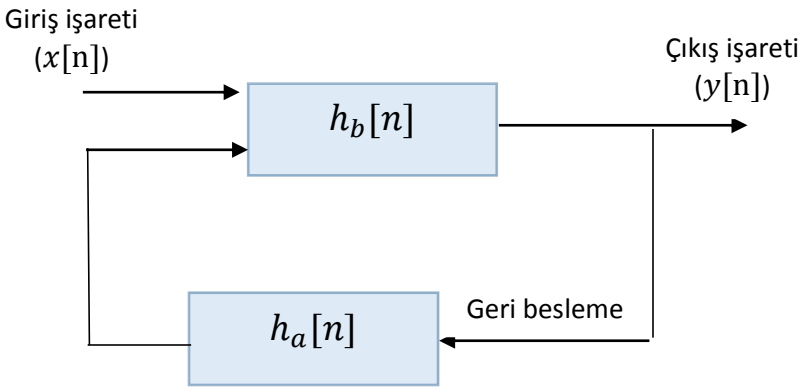
$$\begin{aligned} H\left(z = e^{jw}\right) &= \sum_{k=0}^{N-1} b_k e^{-jwk} \\ &= A(e^{jw}) e^{j\theta(w)} \end{aligned} \tag{2.26}$$

şeklinde bulunabilmektedir. $A(e^{jw})$ filtrenin genlik cevabı, $\theta(w)$ ise filtrenin faz cevabı olarak ifade edilmektedir (Kayran ve Ekşioğlu 2004). Sayısal FIR filtrelerin kullanımından sağladığı avantajlardan bazıları aşağıda verilmektedir (Kayran ve Ekşioğlu 2004, Shenoi 2005).

- FIR filtreler, önceden belirtilen genlik ve faz cevabı özelliklerini doğrusal faz gösterecek şekilde tasarlanabilmektedirler.
- FIR filtreler tekrarlı ya da tekrarsız şekilde gerçekleştirilebilirler. Tekrarlı olmayan gerçekleştirmede doğrudan konvolüsyon veya hızlı Fourier dönüşümü kullanılırken tekrarlı gerçekleştirmede tarak filtresi ve rezonatör bankası kullanılmaktadır.
- Tekrarsız gerçekleştirmede filtrenin transfer fonksiyonunun kutupları olmadığından dolayı sıfır kutup diyagramında birim çember dışına hiç kutup düşmemektedir. Bu da daima kararlı bir yapı sergilediğini göstermektedir.
- FIR filtrede, kuantalama ve yuvarlatma hataları olmaktadır. Tekrarsız gerçekleştirildiklerinden dolayı (çıkış işareti girişten geri besleme almadığından dolayı) çıkış tekrardan sisteme eklenmemekte ve bu hatalar önemsenmemektedir.
- FIR filtreler keskin kesim frekanslı filtre tasarımlarında katsayı hatalarına daha az duyarlıdır. Bu FIR filtrenin tek modlu hata yüzeyinden kaynaklanmaktadır.

2.3.2. IIR filtreler

Sonsuz dürtü cevabına sahip IIR filtreler geri beslemeli filtreler olarak bilinmektedirler. Sahip oldukları bu geri beslemeli yapı dolayısıyla impuls cevapları sonsuz uzunluktadır. Sayısal IIR filtre geri besleme yapısı Şekil 2.19'da gösterilmektedir. Şekil 2.19'dan görüleceği üzere, geri beslemeli IIR yapısı gerçeklenirken, girişin o andaki ve geçmişteki değerleri pay polinomu katsayılarıyla ($h_b[n]$) ve çıkışın geçmişteki değerleri payda polinomu katsayılarıyla ($h_a[n]$) çarpılmaktadır (Mitra 2001). Şekil 2.20'de yapısı gösterilmiş olan direkt form IIR filtre fark denklemi Denklem (2.27)'deki gibi tanımlanabilir.



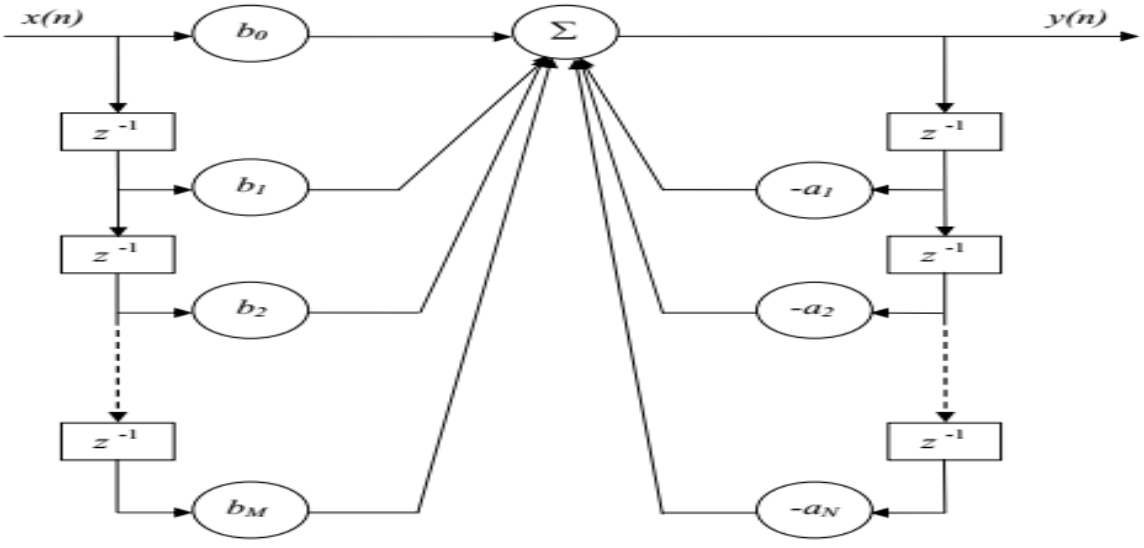
Şekil 2.19. IIR filtre geri besleme yapısı

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k] \qquad (2.27)$$

Buradan,

$$\sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k] \qquad a_0 = 1 \qquad (2.28)$$

elde edilir. x[n] ve y[n] sırasıyla filtrenin giriş ve çıkış işaretlerini, N (≥ M) ise filtrenin derecesini tanımlamaktadır.



Şekil 2.20. Direkt form IIR filtre yapısı (Çetinkaya 2010)

IIR filtrenin fark denkleminden Denklem (2.29)'daki gibi transfer fonksiyonu elde edilir.

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} \qquad a_0 = 1 \tag{2.29}$$

Burada, a_k ve b_k filtre katsayılarını temsil etmektedir. Filtrenin frekans cevabı,

$$H\left(z = e^{jw}\right) = \frac{\sum_{k=0}^{M} b_k \cos(kw) - j\sum_{k=0}^{M} b_k \sin(kw)}{\sum_{k=0}^{N} a_k \cos(kw) - j\sum_{k=0}^{M} a_k \sin(kw)}$$
$$= |H\left(e^{jw}\right)| e^{j\theta(w)} \tag{2.30}$$

şeklinde elde edilir. Burada $|H\left(e^{jw}\right)|$ genlik cevabı, $\theta(e^{jw})$ faz cevabı, w radyan cinsinden normalize frekansı ifade etmektedir (Shenoi 2005). Sayısal filtre tasarımında kararlılık, sayısal filtrelerin pratikte kullanımı açısından önemli bir kavramdır. Bu nedenle tasarlanan sayısal IIR filtrelerin kararlı olmasına ihtiyaç duyulur. IIR filtrelerin kararlı olabilmesi için tüm kutuplarının sıfır-kutup diyagramında birim çember içerisinde kalmış olması gerekmektedir. IIR filtrelerin kutupları karmaşık düzlemde birim çember üzerinde veya dışında ise IIR filtre kararsız olarak nitelendirilir. IIR filtreler giriş içinde geçmişteki çıktıları kullandığından uygun tasarımlarda bu sorunu ortadan kaldırmasına rağmen yine de kararlılık dikkat edilmesi gereken bir kavramdır. IIR filtre tasarlarken karşılaşılabilecek bir başka problem ise, IIR filtrelerinin hata yüzeyi sıklıkla birden çok yerel minimum noktası içerdiğinden dolayı en iyi çözümü temsil eden küresel minimum noktasını bulurken zorluklarla karşılaşılmasıdır. Sayısal IIR filtrelerin sağlanılan avantajlardan bazıları şunlardır (Kayran ve Ekşioğlu 2004, Shenoi 2005):

- İstenilen frekans karakteristiğindeki sayısal filtreyi daha az hesaplama ve hafıza kullanarak tasarlayabilmektedirler.
- Aynı filtreleme işlemi için FIR filtreye göre daha az katsayıya ihtiyaç duymaktadır.
- Aynı dereceye sahip IIR filtrenin frekans seçiciliği FIR filtreye göre daha iyidir.
- Klasik analog sistemlere daha iyi bir yaklaşım sergilemektedir.

- Daha az gecikme operatörü kullanarak FIR filtrelerle aynı genlik cevabına ulaşabilmektedir. Diğer bir deyişle, daha düşük filtre derecesiyle FIR filtrelerle benzer sonuçlara ulaşabilmektedir.

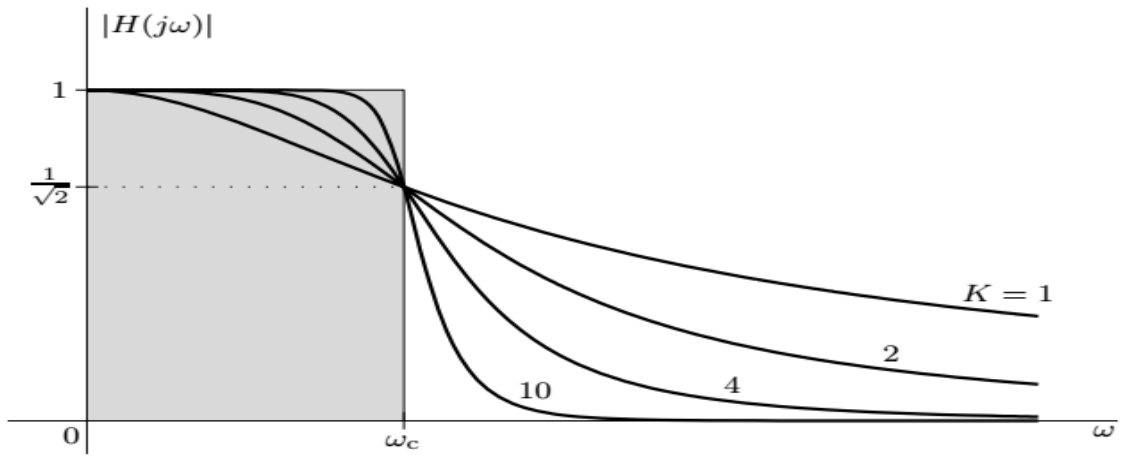
2.3.3. Sayısal filtre tasarım yaklaşımları

2.3.3.1. Butterworth yaklaşımı

Bu yaklaşım yönteminde, tasarlanacak filtrenin genlik cevabının geçirme bandında en yüksek doğruluğa sahip olması istenir. Bundan dolayı geçirme bandında herhangi bir dalgalanma gözlemlenmez. Bu filtre türünün geçiş bandı, aynı dereceli Chebyshev ve eliptik filtrelere göre daha geniş bir bölgeyi kapsamaktadır. K. dereceden Butterworth alçak geçiren filtrenin frekans cevabı Denklem (2.31)'de verilmiştir.

$$|H(j\omega)| = \frac{1}{\sqrt{1+(\frac{\omega}{\omega_c})^{2K}}} \tag{2.31}$$

Denklem (2.31)'de w_c filtrenin kesim frekansı olarak alınmaktadır. $\omega = 0$ noktasında $|H(j0)|$ birim kazanç sağlanmaktadır. $\omega = \omega_c$'de ise kazanç $1/\sqrt{2}$ veya -3 dB olmaktadır. Şekil 2.21'de farklı filtre derecelerine (K) karşılık, alçak geçiren Butterworth filtrenin genlik cevap grafiği verilmektedir.



Şekil 2.21. Farklı dereceden Butterworth alçak geçiren filtrelerin genlik cevabı (Shenoi 2005)

Şekil 2.21'de birinci dereceden onuncu dereceye kadar Butterworth alçak geçiren filtrenin genlik cevabı görülmektedir. Şekilden de görüldüğü üzere, filtrenin derecesi arttıkça filtre ideale daha yakın bir davranış sergilemektedir. Butterworth filtrenin genlik cevabı monoton olarak azalır ve genlik cevabının ilk $2K-1$ türevleri $\omega = 0$ noktasında sıfırdır. Bundan dolayı Butterworth filtrenin cevabı maksimum düz (maximally flat) olarak adlandırılır (Shenoi 2005).

2.3.3.2. Chebyshev yaklaşımı

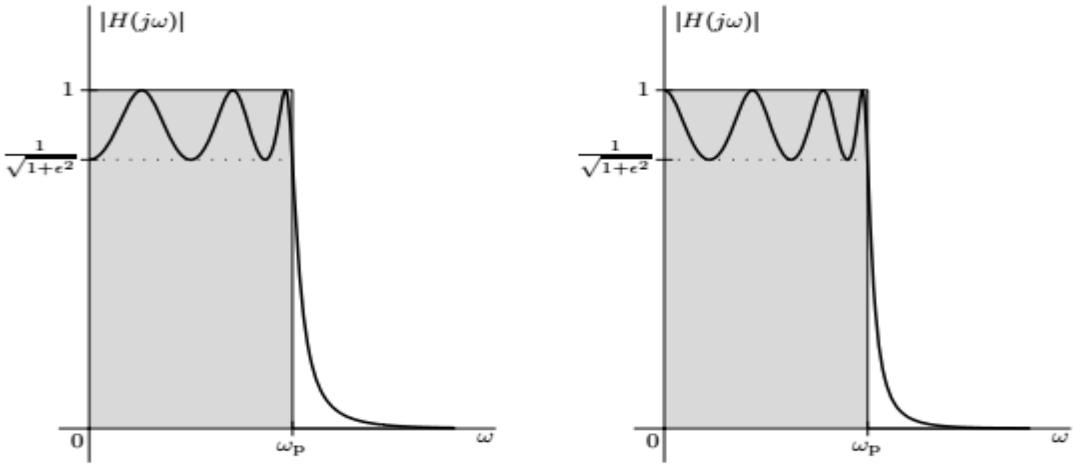
Chebyshev filtre türünün farklı tipleri mevcuttur. Burada çalışmamızda da kullanılan, dalgalanmayı geçirme bandında gösteren Chebyshev Tip 1 filtre ele alınacaktır. Chebyshev Tip 1 alçak geçiren filtrenin genlik cevabı Denklem (2.32)'deki gibidir.

$$|H(j\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 C_k^2(\frac{\omega}{\omega_p})}} \tag{2.32}$$

Ɛ dalgacık parametresidir, geçirme bandındaki dalgacıkların karakteristiğini belirler. $C_k(x)$ k. dereceden Chebyshev polinomu olarak adlandırılır ve Denklem (2.33)'te verilmektedir.

$$C_k(x) = \cos\left(k\cos^{-1}(x)\right) \qquad \text{veya} \qquad C_k(x) = \cos h\left(k\cos^{-1}(x)\right) \tag{2.33}$$

Klasik olarak, Denklem (2.33)'ün kosinüs formu $|x|\leq 1$ ve hiperbolik kosinüs formu $|x|\geq 1$ durumunda kullanılır. x değişkeninin $-1 \leq x \leq 1$ aralığındaki değerlerine karşılık, $C_k(x)$ -1 ile +1 arasında salınım yapar. Bundan dolayı Chebyshev polinomu, eş dalgacıklı (equripple) fonksiyonlar olarak bilinirler. Şekil 2.22'de altı ve yedinci dereceden Chebyshev alçak geçiren filtrelerin genlik cevap grafiği gösterilmektedir. Geçirme bandındaki maksimum kazancın minimum kazanca oranı tepeden tepeye dalgalanma (peak-to-peak ripple) olarak adlandırılır. Şekil 2.2'den görüldüğü üzere filtrenin kazancı en çok 1 ile $1/\sqrt{1+\varepsilon^2}$ arasında değişmektedir. Kesim frekansına gelindiğinde kazanç geçirme bandındaki en düşük değerini almaktadır (Shenoi 2005).



Şekil 2.22. Altıncı ve yedinci dereceden Chebyshev alçak geçiren filtrelerin genlik cevabı (Shenoi 2005)

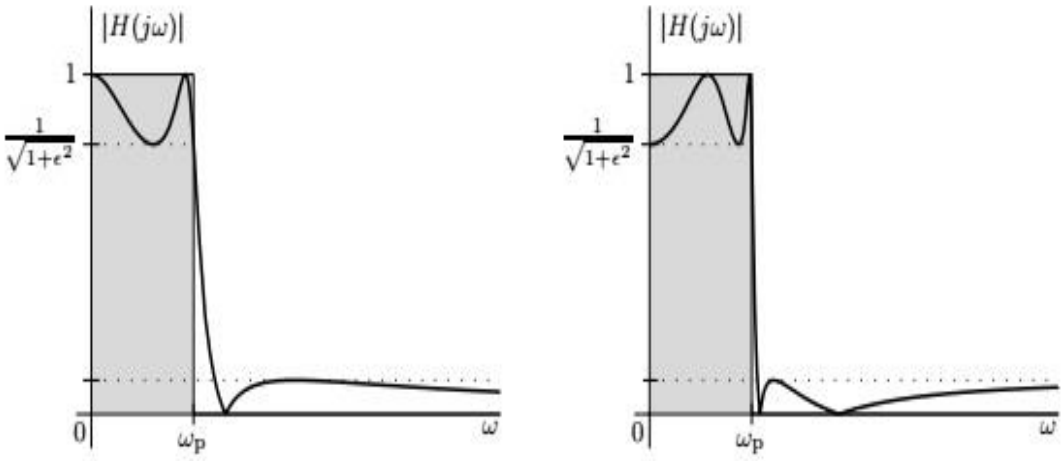
2.3.3.3. Eliptik yaklaşımı

Chebyshev filtreler Butterworth filtrelere nazaran geçirme veya durdurma bandında dalgalanmaya izin verdiklerinden dolayı daha düşük geçiş bandına sahiptirler. Eğer hem geçirme hem de durdurma bandında dalgalanmalara izin verilirse, daha iyi bir geçiş bandı performansı yakalanabilmektedir. Bu tür yaklaşım eliptik yaklaşım (Cauer) olarak adlandırılmaktadır. Aynı dereceden filtrelerde, daha önce bahsedilmiş olan yaklaşımlara göre geçiş bandı daha kısadır. Eliptik filtrenin genlik cevabı Denklem (2.34)'te verilmektedir.

$$|H(j\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 R_k^2(\frac{\omega}{\omega_p},\xi)}} \tag{2.34}$$

Burada, $R_k\ (\frac{w}{w_p},\xi)$ fonksiyonu $k.$ dereceden eliptik rasyonel fonksiyonu olarak adlandırılmaktadır. Şekil 2.23'te üçüncü ve dördüncü dereceden eliptik alçak geçiren filtrelerin genlik cevap grafiği gösterilmektedir. Şekil 2.23'ten görüldüğü üzere eliptik filtreler geçirme bandında 1 ile $1/\sqrt{1+\varepsilon^2}$ arasında dalgalanma sınırlarına sahiptirler.

Eliptik yaklaşım, dar geçiş bandı istenen uygulamalarda tercih edilen bir yaklaşımdır (Shenoi 2005).



Şekil 2.23. Üçüncü ve dördüncü dereceden eliptik filtrelerin genlik cevabı (Shenoi 2005)

3. EVRİMSEL ALGORİTMALAR

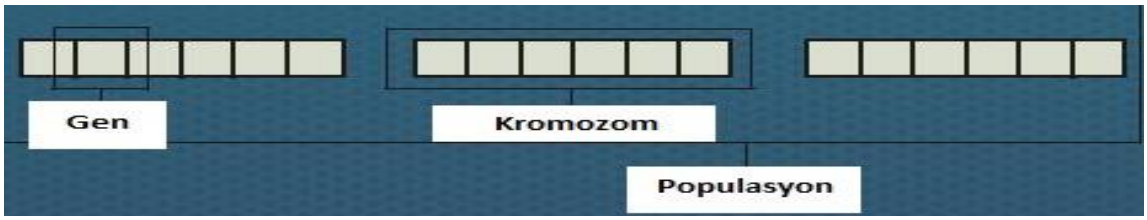
Evrimsel algoritmalar fen, bilgisayar, elektrik ve elektronik bilimlerinde sıklıkla kullanılan bir problem çözme yöntemidir. Evrimsel algoritmalar en iyi çözümün bulunması garanti olmamasına rağmen, makul bir sürede bir çözüm elde edeceklerini garanti etmektedirler (Reeves 1993, Simon 2013). Çözülen problemin türüne bağlı olarak bulunan çözüm genellikle en iyiye yakın olup, hızlı ve kolay bir şekilde bu çözüme ulaşılmaktadır (Yang 2010). Bu bölümde, bu çalışmada kullanılan evrimsel algoritmalar; genetik, diferansiyel gelişim ve parçacık sürü optimizasyon algoritmalarının işlem adımları ve çözüme ulaşırken kullandığı metotlar detaylı bir şekilde anlatılacaktır.

3.1. Genetik Algoritma

3.1.1. Giriş

Genetik algoritma (Holland 1975), doğada gözlemlenen evrimsel süreçten esinlenen, en uygun çözümü (optimal) arayan bir en iyileme yöntemidir. Bir veya birden çok boyuta sahip karmaşık problemlerin arama uzayında, problemin bütünsel çözümünü ararken en iyinin hayatta kalması ilkesine dayanır. Bir çözümün iyi ya da kötü olduğuna karar verebilmek için hedef fonksiyonlarının uygunluk değerlerinden faydalanır. Genetik algoritmalar popülasyon diye nitelendirilen, problemlerin farklı olası çözümlerinden oluşan bir çözüm kümesi üretir. Böylelikle, popülasyon içindeki çözüm adayları, arama uzayında birden çok noktayı tarayarak bütünsel çözüme ulaşma olasılığını arttırabilmektedir. Çözüm kümesindeki çözümler birbirinden tamamen bağımsızdır ve popülasyon içindeki çözüm adaylarının hep daha iyiye gitmesi amaçlanmaktadır (Reeves 1993).

Genetik algoritmalar evrim teorisinden esinlendiğinden dolayı, probleminin çözüm kümesini temsil eden popülasyon, kromozom adı verilen problemin olası çözümlerini temsil eden vektörlerden oluşmaktadır. Kromozom içindeki her bir elemana (parametreye) gen adı verilir. Popülasyondaki kromozomlar, evrimsel süreç içinde genetik algoritma işlemcileri tarafından belirlenirler. Genetik algoritma popülasyon yapısı Şekil 3.1'de verilmektedir.



Şekil 3.1. Genetik algoritmada popülasyon yapısı

Genetik algoritmada genler yaygın olarak ikili sayı sistemi elemanları olan 0-1 rakamları ile kodlanmaktadır. Başlangıç popülasyonunda probleme ilişkin sınırlar dahilinde rastgele genler kodlanarak kromozomları oluşturmaktadır. Problemin olası çözümlerinin kromozomlarda kodlanması, popülasyonun büyüklüğü, gen sayısı problemden probleme değişiklik göstermektedir. Kromozomların ikili kodlanmasıyla popülasyonun elde edilmesine bir örnek Şekil 3.2.'de verilmektedir.

	Gen_1	**Gen_2**	**Gen_3**	**Gen_4**	**Gen_5**	**Gen_6**
$Kromozom_1$	1	0	1	1	0	1
$Kromozom_2$	0	0	1	0	1	0

Şekil 3.2. Popülasyonun elde edilmesi

Şekil 3.2'de genetik algoritmada iki kromozoma sahip popülasyon görülmektedir. Problemin karmaşıklığı arttıkça popülasyondaki kromozom sayısı değişmekte ve problemin içerdiği değişkenlere bağlı olarak gen sayısı artmakta veya azalmaktadır. Genetik algoritmada verilen başlangıç kromozomların ve genlerin sayısı, evrimsel süreç boyunca sabit kalmaktadır.

3.1.2. Çalışma prensibi

Genetik algoritmada, kromozomların problemin çözümündeki başarısına karar vermesindeki en önemli faktör kromozomların evrimsel süreç boyunca daha iyiye gelişimidir. Bu gelişim sürecinde, kromozomların problem için uygun bir çözüm olup olmayacağına karar verilmesini sağlayan bir hedef fonksiyonu mevcuttur. Denklem (3.1)'de hedef fonksiyonu genel haliyle verilmektedir.

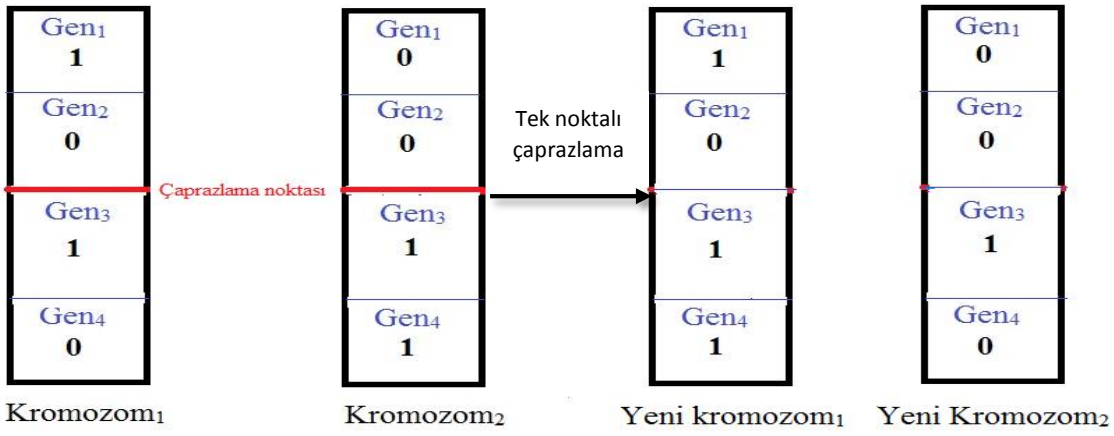
$$HF = f(x_1, x_2 \ldots, x_n) \quad (3.1)$$

Hedef fonksiyonunda x, n tane gene sahip kromozomu temsil etmektedir. *HF* hedef fonksiyonundan dönen değere göre yüksek uygunluk değerine sahip olan kromozomlar vasıtasıyla yeni bireyler oluşturularak popülasyona eklenir. Kromozomların çözüm arama sürecinde, daha iyi çözümleri aramasını sağlamak amacıyla genetik operatörler kullanılır. Kullanımı en çok tercih edilen operatörler; çaprazlama, mutasyon, seçilim ve elitizm işlemleridir (Colin ve Jonathan 2003).

3.1.2.1. Çaprazlama işlemi

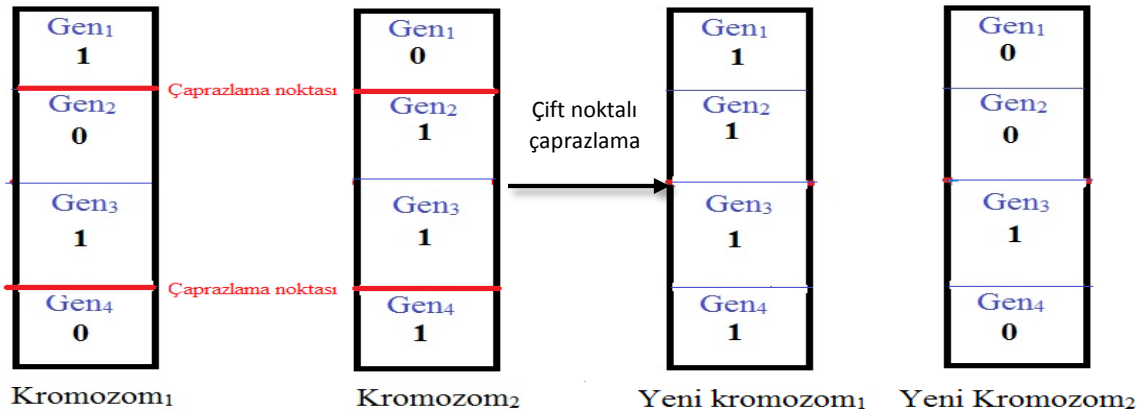
Çaprazlama işlemi, en genel tanımıyla seçilen iki farklı kromozom arasında yapılan gen değiş-tokuş işlemidir. Bu işlem sonucunda iki yeni yavru kromozom ortaya çıkmaktadır. Bu yeni kromozomlar bir sonraki nesle aktarılırlar. Çaprazlama işleminde uygun bir çaprazlama noktası seçildiğinde problemin çözümüne uygun bir öncekine nazaran daha iyi kromozomlar elde edilmektedir. Fakat problemin türüne bağlı olarak uygun bir çaprazlama noktası için genel bir tanım olmadığından çaprazlama noktası rastgele olarak seçilmektedir. Genetik algoritmada, çaprazlamanın sıklığını belirleyen bir çaprazlama oranı mevcuttur. Eğer bu oran sıfir ise üretilen yavru kromozomlar üretildikleri kromozomların aynısı olmakta ve evrimsel süreçte kromozomlar arasında çeşitlilik sağlanamadığından daha iyi çözümler üretilememektedir. Genetik algoritmada çaprazlama oranı genellikle 0,7 olarak seçilmektedir. Çaprazlama işleminde sıklıkla iki farklı yöntem kullanılmaktadır.

Tek noktalı çaprazlama yöntemi, seçilen kromozom çifti arasından ilk ve son genler dışındaki aradaki genler arasında rastgele olarak bir çaprazlama noktası seçilir. Bu noktadan itibaren kromozom çiftleri karşılıklı olarak gen değiş tokuşu yaparak yeni yavru kromozomlar oluşturulurlar. Oluşturulan yavru kromozomlar ilgili problemin yeni çözüm adaylarını temsil etmektedir. Tek noktalı çaprazlama yöntemi Şekil 3.3'te gösterilmektedir.



Şekil 3.3. Tek noktalı çaprazlama yöntemi

Çift noktalı çaprazlama yöntemi, seçilen kromozom çifti arasından ilk ve son genler dışındaki aradaki genler arasında rastgele olarak iki adet çaprazlama noktası seçilir. Çaprazlama işlemi seçilen bu iki nokta arasındaki genlerin yer değiştirmesidir. Çift noktalı çaprazlama yapılabilmesi için seçilen kromozomların en az üç genden oluşması gerekmektedir. Şekil 3.4'te iki noktalı çaprazlama yöntemi gösterilmektedir.

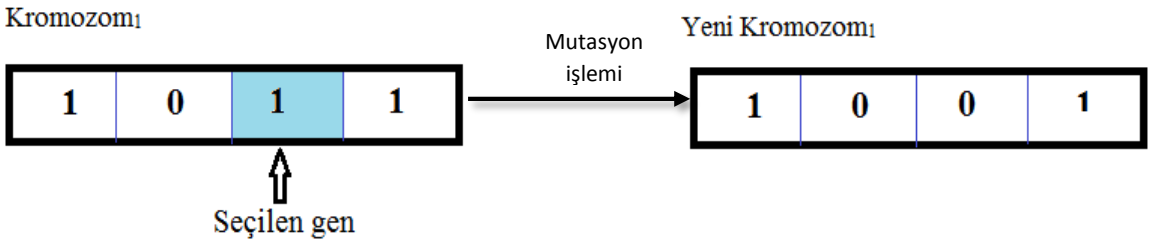


Şekil 3.4. Çift noktalı çaprazlama yöntemi

3.1.2.2. Mutasyon işlemi

Evrimsel süreçte mutasyon, belirli bir genin doğal olmayan yollardan değişimini ifade etmektedir. Eğer popülasyonda problemin olası çözümlerini temsil eden kromozomlar gereksinim duyulan tatmin edici bir çözüme ulaşamıyorsa, gen havuzundan yeni

kromozomlar üretilme ihtiyacı duyulmaktadır. Mutasyon işlemi bu görevi sağlayarak, problemin herhangi bir yerel minimum noktasına takılmasını engelleyebilmektedir. Bu işlem kromozomlar arasında çeşitliği arttırarak, daha önceki çözüm aramalarında elde edilmemiş olan yeni çözüm adayı kromozomların oluşumunu sağlamaktadır. Fakat genetik algoritmada bu işleme sıklıkla başvurulmamaktadır. Eğer bu işlem sıklıkla tekrarlanırsa, çözümler sürekli değişime uğrayacak ve üretilen yeni çözümlerin en iyi çözümden uzaklaşma olasılığı artacaktır. İkili sayı sistemi ile kodlandığında mutasyon işlemi 0 olan genin 1 ya da 1 olan genin 0 olması anlamına gelmektedir. Örnek mutasyon işlemi Şekil 3.5'te gösterilmektedir.



Şekil 3.5. Mutasyon işlemi

3.1.2.3. Seçilim işlemi

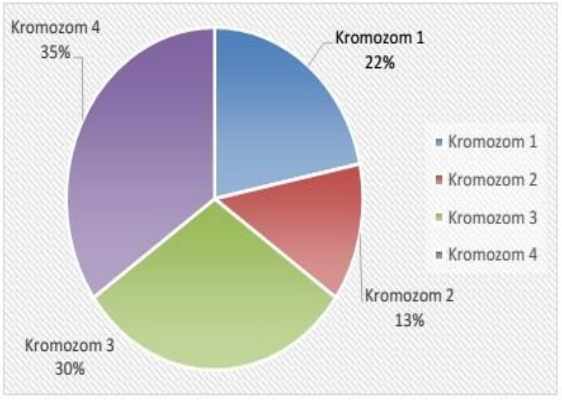
Genetik algoritmada seçilim işlemi, evrimsel süreç örnek alınarak bir nesilden yeni bir nesle geçiş ve bu nesildeki bulunacak bireylerin belirlenmesi olarak tanımlanmaktadır. Daha iyi çözümlere ulaşmak amacıyla yeni nesillere aktarılacak kromozomların seçimi oldukça önem arz etmektedir. Bu seçilim işlemi için genetik algoritmada sıklıkla aşağıdaki yöntemler kullanılmaktadır.

Turnuva seçimi yönteminde, popülasyondan önceden belirlenen sayıda rastgele kromozom seçilir. Seçilen kromozomların problemi çözmedeki başarısı uygunluk değerlerine bağlı olarak belirlenmektedir. Belirlenen p olasılığına göre popülasyondaki en iyi kromozom seçilmektedir. Belirlenen olasılığın artması popülasyondaki en iyi kromozomun seçim olasılığını arttırmakta, bu değerin 1 olması daima en iyi kromozomun seçiliminin yapılacağı anlamına gelmektedir. Bu yöntemde popülasyondaki çeşitliliği arttırmak amacıyla önceden seçilen bir kromozomun tekrar seçilmesi engellenerek, olası çözümleri

temsil eden kromozomların çözüm arama sürecinde tek bir bölgede takılmasının önüne geçilmeye çalışılmaktadır. Gruptan seçilemeyen kromozomlar başka bir gruba dahil olup yeniden seçilme şansı arayarak popülasyondaki tüm kromozomlara seçilme şansı tanınmaktadır.

Rulet çemberi seçimi yönteminde, kromozomlar uygunluk değerleriyle orantılı bir olasılıkta rulet çemberine yerleştirilmektedir. Bu oran, popülasyondaki tüm kromozomların uygunluk değerleri toplanarak her bir kromozomun uygunluk değerine bölünmesiyle bulunur. Kromozomların rulet çemberinde kapsayacağı alanın yüzdesel değeri bu oran tarafından temsil edilmektedir. Rulet çemberi oluşturulduktan sonra 0-100 aralığında bir sayı seçilir ve bu sayının rulet tekerleği üzerinde denk gelen kromozom seçilir. Bu işlem rulet çemberini döndürme işlemi olarak tanımlanmaktadır. Daha iyi uygunluk değerine sahip kromozomlar rulet çemberinde daha çok yer kaplayacağından seçilme olasılıkları daha yüksektir. Şekil 3.6'da rulet çemberinin oluşumu ve ilgili kromozomların rulet çemberine yerleşimi gösterilmektedir.

Kromozom	Uygunluk Değeri	Rulet Oranı
Kromozom 1	25	22%
Kromozom 2	15	13%
Kromozom 3	35	30%
Kromozom 4	40	35%
Toplam	115	100%



Şekil 3.6. Rulet çemberi yöntemi

Şekil 3.6'dan görülebileceği üzere, problemin dört farklı olası çözümlerini temsil eden kromozomlar hedef fonksiyonundan kendi uygunluk değerlerini döndürmektedirler. Tüm kromozomlar için toplam bir uygunluk değeri ve buna bağlı olarak her bir kromozomun maksimizasyon problemi için rulet oranı belirlenmektedir. Bu oran kromozomların rulet tekerleğinde kapsayacağı alanı belirtmektedir.

3.1.2.4. Elitizm işlemi

Genetik algoritmada, genetik operatörler vasıtasıyla yapılan işlemler sonucunda üretilen yeni kromozom bazı durumlarda üretildiği kromozomdan probleme daha iyi çözüm sunamamaktadır. Bu durumda yeni üretilen daha kötü çözümü temsil eden kromozomun, üretildiği daha iyi çözüme sahip olan kromozomun yerini alması iyi kromozomun kaybolması anlamına gelmektedir. Elitizm bu durumu engellemek için kullanılmaktadır. Popülasyondaki belirli bir elitizm oranı kadar en iyi olası çözüme sahip kromozomların değişikliğe uğratılmadan sonraki nesile aktarılarak korunması sağlanmaktadır. Bu, çözümün performansının sonraki nesillerde azalmasının önüne geçmektedir. Diğer yandan, popülasyonun elit kromozom veya kromozomlar dışında kalan kromozomlar seçilim, çaprazlama ve mutasyon operatörlerine tabi tutularak, evrimsel süreçte çözümü geliştirmeye devam etmektedir. Bu gelişim sürecinde üretilen yeni kromozomlar seçilen elit kromozomdan daha iyi olmaması durumunda aynı elit kromozom sonraki nesile aktarılarak en iyi kromozomun devamlılığı sağlanmaktadır.

3.1.3. Algoritma adımları

Genetik algoritmanın çözüm adımları aşağıdaki gibi özetlenebilir (Colin ve Jonathan 2003, Yang 2010):

i. Olası çözümleri temsil eden başlangıç kromozomlarını içeren popülasyon oluşturulur.

ii. Popülasyondaki tüm kromozomların uygunluk değerleri hesaplanır.

iii. Kromozomlar seçilme, çaprazlama ve mutasyon işlemlerine tabi tutularak yeni kromozomlar üretilir.

iv. Yeni üretilen kromozomların uygunluk değerleri hesaplanır.

v. Mevcut popülasyondaki elitizm oranındaki en iyi kromozomlar korunarak, yeni nesil oluşturulur.

vi. Yeni nesildeki tüm kromozomlar, mevcut kromozomlar ile yer değiştirerek yeni popülasyon elde edilir.

vii. Algoritmanın sonlandırma kriteri sağlanıyorsa durdurulur, sağlanmıyorsa iii. adımdan devam edilmelir.

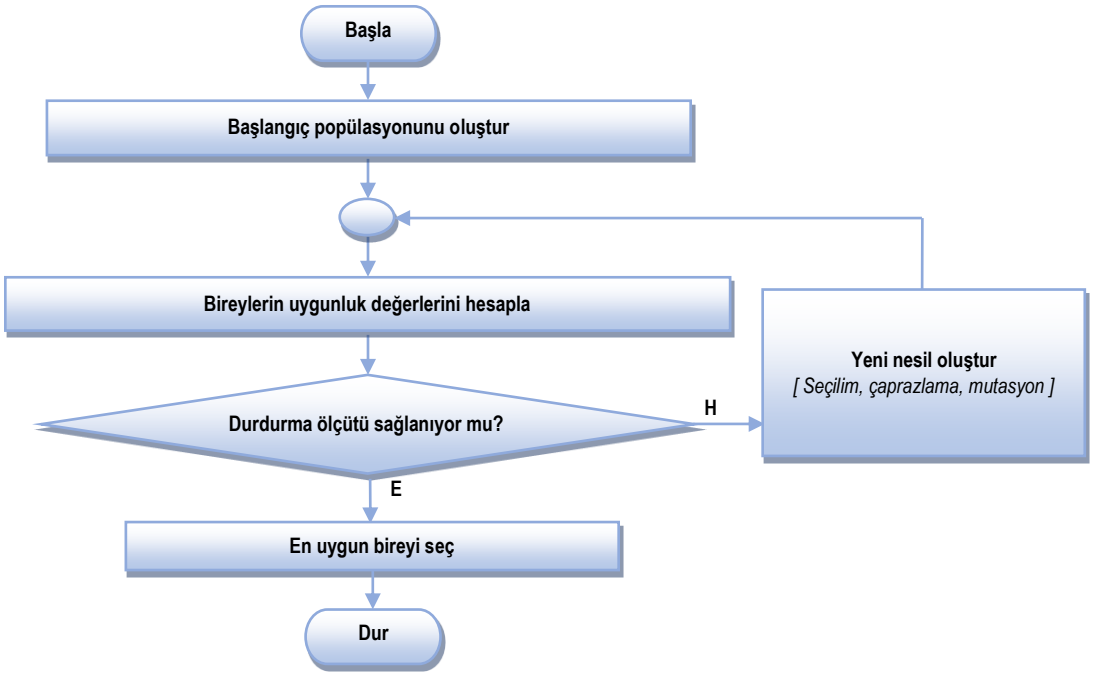
Genetik algoritmanın sözde kodu Şekil 3.7'de verilmektedir (Colin ve Jonathan 2003).

Başla
    Başlangıç parametrelerini belirle.
    Başlangıç populasyonunu oluştur.
Durdurma kriteri sağlanıncaya kadar
{
        Kromozomların uygunluk değerlerini hesapla.
        Popülasyona seçilim işlemi uygula.
        Kromozomlara çaprazlama işlemi uygula.
        Kromozomlara mutasyon işlemi uygula.
        Yeni popülasyonu oluştur.

}
En iyi kromozomu çözüm olarak al.
Bitir

Şekil 3.7. Genetik algoritmanın sözde kodu

Genetik algoritmanın işlem adımlarını gösteren genel akış diyagramı Şekil 3.8'de gösterilmektedir.

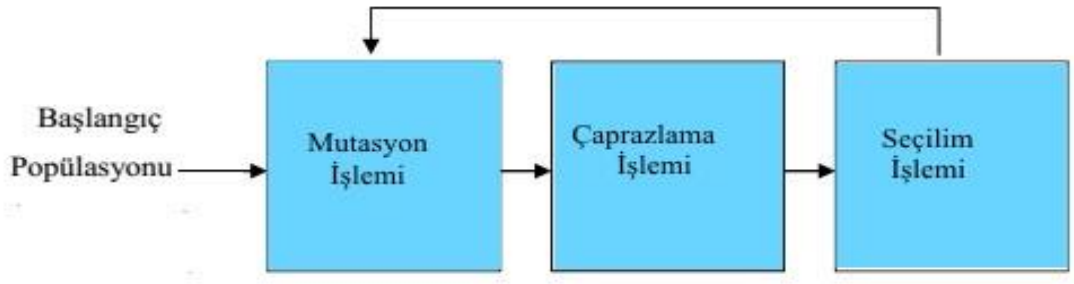


Şekil 3.8. Genetik algoritmanın akış diyagramı

3.2. Diferansiyel Gelişim Algoritması

3.2.1. Giriş

Diferansiyel gelişim algoritması, Price ve Storn tarafından (1995) geliştirilmiş popülasyon tabanlı evrimsel bir algoritmadır. Sürekli verilerin işlendiği çözümlerde iyi sonuçlar vermektedir. DE'nin yerel minimuma hızlı bir şekilde yakınsaması, küresel minimumu bulmadaki başarısı, az sayıda kontrol parametresi kullanması ve gerçek sayılarla çalışması bu algoritmanın kendini öne çıkaran üstün özelliklerinden sayılabilmektedir (Yang 2010). DE diğer evrimsel algoritmalar gibi aynı anda birçok noktada çözüm arama özelliğine sahiptir. Bu özellik algoritmanın çözümü geliştirmesine yardımcı olup, yerel minimum noktalarından kaçmasını sağlamaktadır. Genetik algoritmada kullanılan çaprazlama, seçilim ve mutasyon işlemleri DE'de de kullanılmaktadır. Genetik algoritmadan farklı olarak DE'de değişkenler ikili sayı sistemi ile değil gerçek değerleriyle temsil edilmektedir. DE'nin diğer bir farklı yönü her bir işlemi tüm popülasyona uygulamasıdır. Her bir kromozom tek tek ele alınmakta ve bu kromozomlara mutasyon ve çaprazlama işlemleri uygulanmaktadır. Böylelikle yeni kromozomlar oluşturulabilmektedir. Seçilim işlemi, hedef fonksiyonundan dönen uygunluk değerlerine göre yapılmakta, sonraki nesile aktarılacak bireyler bu yöntemle belirlenmektedir (Storn ve Price 1995, Hrstka ve Kucerova 2004). Evrimsel süreçte yapılan bu işlemlerin genel şeması Şekil 3.9'da verilmektedir.



Şekil 3.9. Diferansiyel gelişim algoritması genel işlem şeması

3.2.2. Çalışma prensibi

DE'de problemin türü ve içerdiği sınırlamalara bağlı olarak bir başlangıç popülasyonu oluşturularak algoritma başlatılır. Bu popülasyon oluşturulurken probleme ait değişken sayısı, DE'de her bir kromozoma ait genler ile temsil edilmektedir. Toplam popülasyon büyüklüğü ise, toplam kromozom sayısını ifade etmektedir. Popülasyon büyüklüğü daima üçten büyük olmalıdır. Bunun sebebi DE'nin mutasyon sürecinde yeni kromozomlar üretebilmesi için mevcut kromozom dışında üç adet daha kromozoma ihtiyaç duymasıdır. Problemin türüne bağlı olarak, tasarım değişkenlerine maksimum ve minimum değerleri atanarak, Denklem (3.2)'deki formülasyona uygun başlangıç popülasyonu oluşturulur (Storn ve Price 1997).

$$X_{i,j}^{0} = L_j + \text{rand} * \left(H_j - L_j\right),\ i = 1, \dots, NP; j = 1, \dots, ND \qquad (3.2)$$

Burada, NP popülasyondaki toplam kromozom sayısını, ND değişken sayısını, X^0 başlangıç populasyonunu, H ve L tasarım değişgenlerinin (kromozomlar) alabileceği üst ve alt sınırları, $rand$ 0-1 aralığında üretilen rastgele sayıyı ifade etmektedir. Başlangıç popülasyonu oluşturulduktan sonra sırasıyla mutasyon, çaprazlama ve seçilim işlemleri tüm popülasyona uygulanır. Kullanıcı tarafından belirlenen durdurma kriteri sağlanıncaya kadar işlemler tekrarlanarak algoritma sonlandırılır. Durdurma kriteri, iterasyon (nesil) sayısı veya hatanın istenilen bir seviyeye düşmesi şeklinde belirlenebilmektedir. Son nesildeki en iyi kromozom, DE'nin ürettiği en iyi çözüm olarak adlandırılmaktadır.

3.2.2.1. Mutasyon işlemi

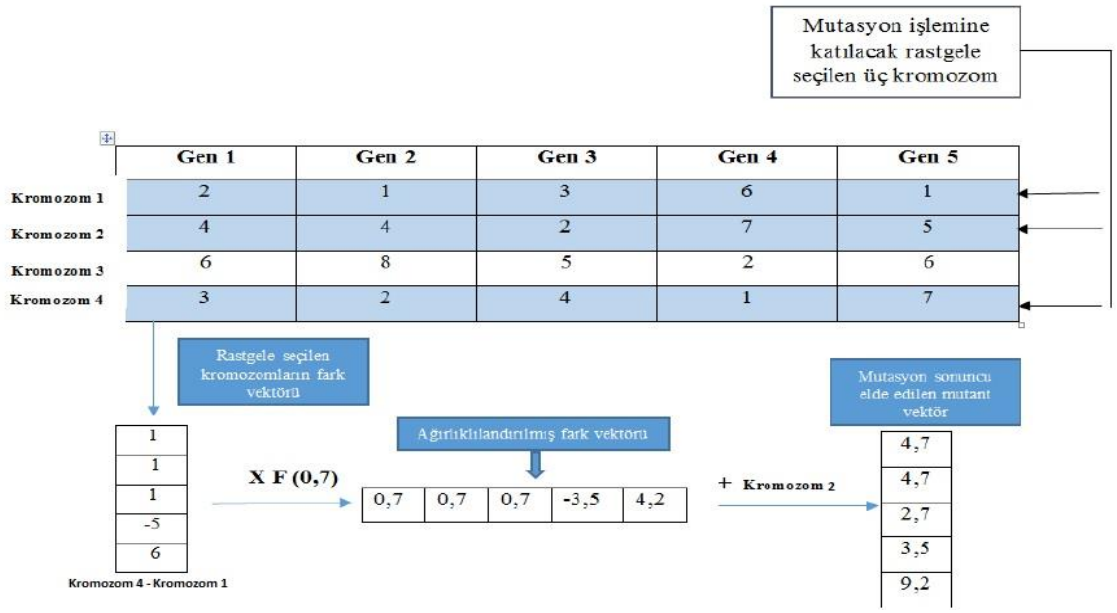
Mutasyon işlemi, mevcut kromozomun bir kısım genleri üzerinde rastgele belirlenmiş şekilde değişiklikler yapılıp önceki kromozomdan farklı bir kromozom oluşturma işlemi olarak tanımlanabilir. Bu işlem genel olarak popülasyonu daha güçlü hale getirerek yeni çözüm bölgelerinin araştırılmasına yardımcı olmaktadır. Mutasyon işlemine tabi tutulacak mevcut kromozom dışında üç adet daha kromozom seçilerek (r_1, r_2, r_3) yeni kromozom oluşturma işlemi yapılmaktadır. Storn ve Price, mutasyon işlemi için altı farklı operatör geliştirmişlerdir. Kullanılan operatöre göre mutasyon işlemi için farklı kromozomlar

seçilerek, yeni mutant kromozom elde edilmektedir. Mutasyon işleminde kullanılan operatörler ve bunlara ait işlemler Çizelge 3.1'de tablolandırılmaktadır (Storn ve Price 1997).

Çizelge 3.1. Mutasyon işleminde kullanılan operatörler

Mutasyon işlemi	**Formülizasyonu**
DE/rand/1	$V_i = X_{r_3} + F * (X_{r_1} - X_{r_2})$
DE/best/1	$V_i = X_i + F * (X_{best} - X_i) + F * (X_{r_1} - X_{r_2})$
DE/rand-to-best/1	$V_i = X_{best} + F * (X_{r_1} - X_{r_2})$
DE/best/2	$V_i = X_{best} + F * (X_{r_1} - X_{r_2}) + F * (X_{r_3} - X_{r_4})$
DE/rand/2	$V_i = X_{r_1} + F * (X_{r_2} - X_{r_3}) + F * (X_{r_4} - X_{r_5})$
DE/rand-to-best/2	$V_i = X_i + F * (X_{best} - X_i) + F * (X_{r_1} - X_{r_2}) + F * (X_{r_3} - X_{r_4})$

Burada, X_r rastgele seçilen kromozomları, X_{best} populasyondaki en iyi kromozomu, X_i mevcut kromozomu, F ölçekleme faktörünü ifade etmektedir. Bu çalışmada DE/rand/1 mutasyon yöntemi kullanılmıştır. Bu yöntemin rastgele oluşturulan popülasyona göre örnek uygulanışı Şekil 3.10'da verilmektedir.



Şekil 3.10. DE/rand/1 yöntemi örnek gösterimi

Şekil 3.10'da verilen örnekte popülasyon, dört adet kromozom ve her bir kromozom beş adet genle temsil edilmektedir. Her bir gen içinde seçilen problemin olası çözümlerini oluşturacak sayılardan meydana gelmektedir. DE/rand/1 yönteminde başlangıçta üç adet rastgele kromozom seçilir. Bu kromozomlardan iki tanesi birbirinden çıkarılarak fark vektörü bulunur. Fark vektörü, bu örnek için 0,7 olarak alınmış, F parametresiyle çarpılarak ağırlıklandırılmış fark vektörü bulunur. Bu vektör ile bu örnek için Kromozom 4'ün temsil ettiği rastgele seçilmiş olan üçüncü bir vektör ile toplanarak mutant vektör elde edilir. Böylece mutasyon sonucu çaprazlamada kullanılacak olan kromozom (mutant vektör) elde edilmiş olur.

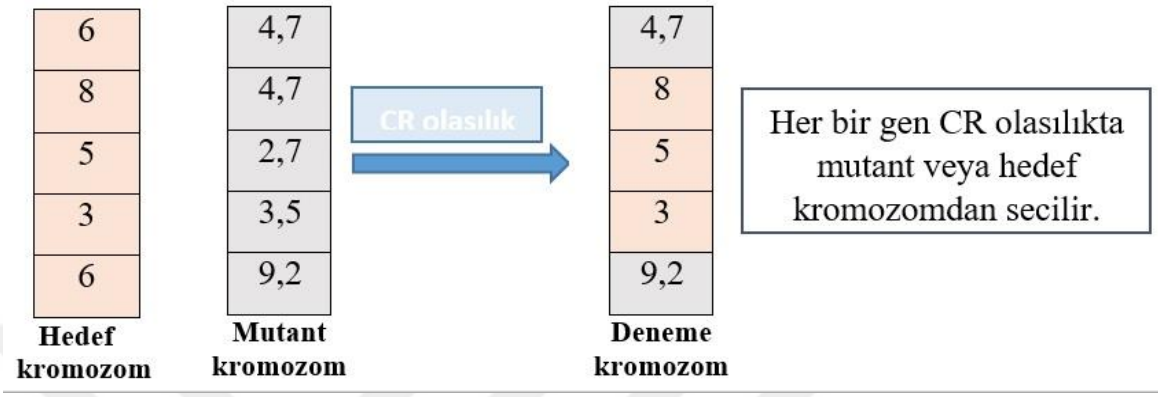
3.2.2.2. Çaprazlama işlemi

DE'de çaprazlama işleminin amacı var olan kromozomlar vasıtasıyla yeni kromozomlar üreterek çözüm uzayında aramanın daha iyi performans vermesine yardımcı olmaktır. Çaprazlama işleminde, mutasyon sonucu elde edilen kromozom ile önceki nesilden elde edilen kromozom çaprazlama işlemine tabi tutularak yeni deneme kromozomu elde edilmektedir. Klasik DE'de ve bu çalışmada kullanılan çaprazlama binom formunda çaprazlamadır. Çaprazlama işlemi Denklem (3.3)'te görülmektedir (Storn ve Price 1997).

$$U_{i,j}^{k+1} = \begin{cases} V_{i,j}^{k+1} \; eğer \; rand_j(0,1) \; \leq CR \; ya \; da \; j = j_{rand} \\ X_{i,j}^{k} \; diğer \; durumda \end{cases} \tag{3.3}$$

DE'de genetik algoritmadan farklı olarak, eşit olasılık yerine CR olasılığı mevcuttur. U_i deneme kromozomuna ait her bir gen, 0 ile 1 arasında üretilen rastgele sayıya bağlı olarak, CR olasılıkla mutant kromozomdan (V_i), $1 - CR$ olasıkla mevcut kromozomdan (hedef kromozom, X_i) seçilir. Amaç belirlenen oranda genin mutant kromozomdan alınıp yeni kromozomlar üretilmesidir. j_{rand} parametresi en az bir tane genin mutant kromozomdan alınmasını garanti etmek amacıyla bulunmaktadır. Rastgele seçilen j_{rand} noktasındaki gen CR olasılığına bakılmaksızın mutant kromozomdan seçilmektedir (Storn ve Price 1995).

Şekil 3.11'de çaprazlama işleminin örnek uygulanışı verilmektedir.



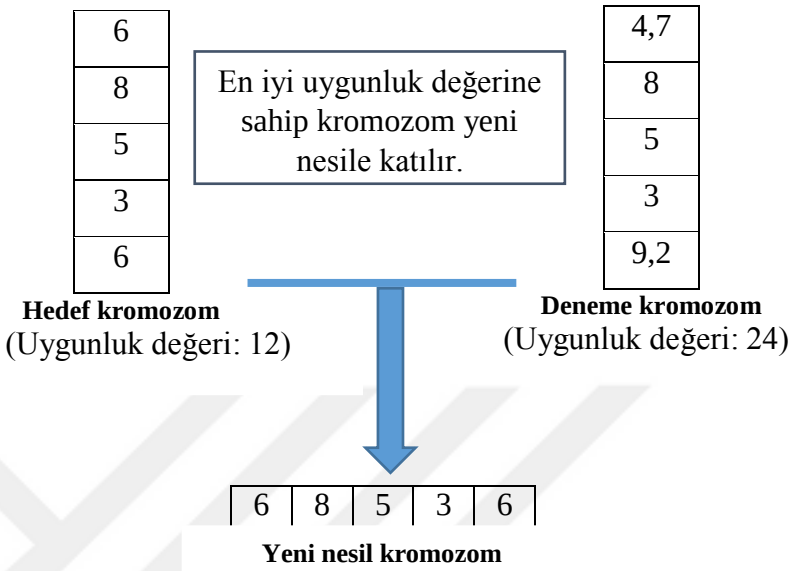
Şekil 3.11. Çaprazlama işlemi örnek gösterimi

3.2.2.3. Seçilim işlemi

Seçilim işleminde temel amaç yeni nesile katılacak bireylerin belirlenmesidir. Popülasyonda bulunan hedef kromozom ile mutasyon ve çaprazlama işlemleri sonucu elde edilen deneme kromozomu, hedef fonksiyondan dönen uygunluk değerleri vasıtasıyla karşılaştırma yapılarak yeni nesil kromozomu seçimi yapılır. Hangi kromozomun uygunluk değeri daha iyiyse (çözümü geliştirmede daha iyi performans sergiliyorsa) o kromozom yeni nesili oluşturacak kromozomlar arasına katılır. Bu işlem, Denklem (3.4)'teki gibi formülize edilmektedir (Storn ve Price 1997).

$$\forall\, i \leq NP\colon\ X_i^{k+1} = \begin{cases} U_i^{k+1}\ eğer\ f(\,U_i^{k+1}) \leq f(X_i^k) \\ X_i^k\ diğer\ durumda \end{cases} \tag{3.4}$$

Burada, NP popülasyon sayısını, X_i^k hedef kromozomu, X_i^{k+1} yeni nesilin $i.$ kromozomunu, U_i^{k+1} deneme kromozomunu, $f(\,U_i^{k+1})$ deneme kromozomunun uygunluk değerini ve $f(X_i^k)$ hedef (mevcut) kromozomun uyguluk değerini belirtmektedir. Şekil 3.12'de herhangi bir minimizasyon problemi için rastgele sayılarla oluşturulan kromozomlar ve uygunluk değerlerine ilişkin örnek seçilim işlemi temsil edilmektedir.



Şekil 3.12. Örnek seçilim işlemi

3.2.3. Algoritma adımları

Diferansiyel gelişim algoritmasının işleyişin adımları aşağıdaki gibi özetlenmektedir (Storn ve Price 1995, 1997).

i. Tüm parametre uzayını kapsayacak şekilde olası çözümleri temsil eden kromozomları içeren başlangıç popülasyonu oluşturulur.

ii. Popülasyondaki tüm kromozomların uygunluk değeri hesaplanır.

iii. Hedef kromozom dışında üç kromozom seçilerek, F ölçekleme çarpanı yardımıyla mutasyon işlemine tabi tutulur.

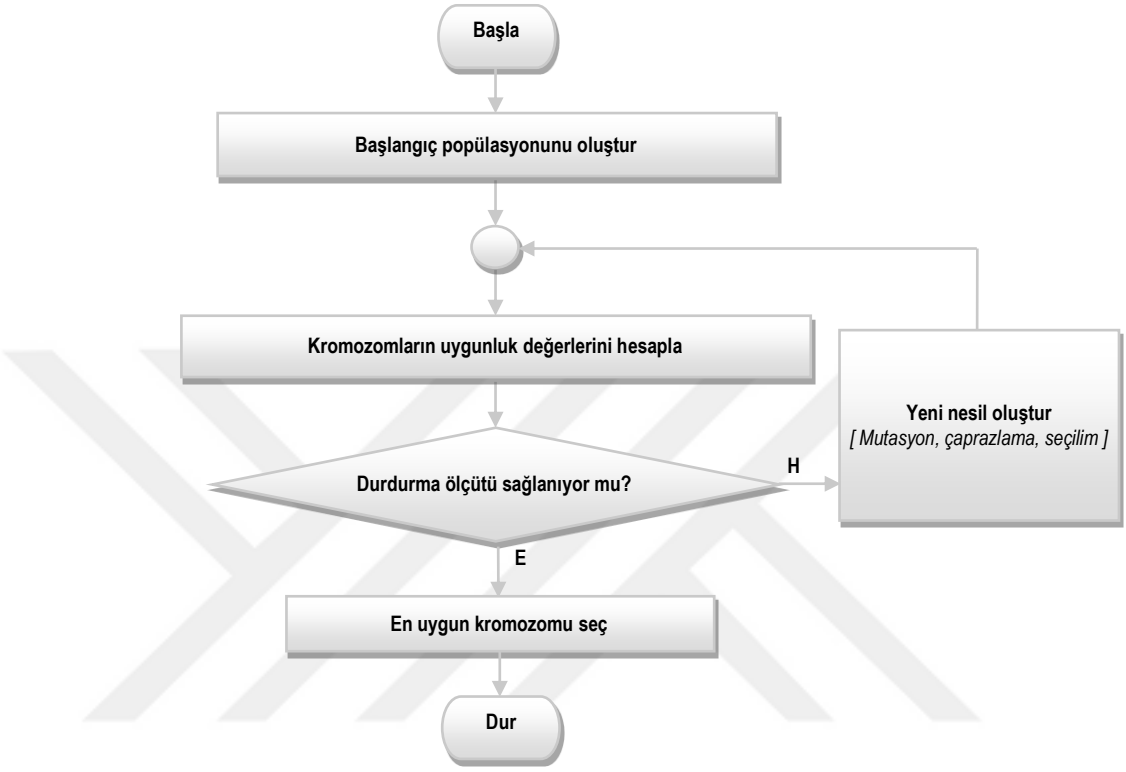
iv. Mutasyona uğramış kromozomdan CR oranında gen, hedef kromozomdan $1 - CR$ oranında gen alınarak deneme kromozomu oluşturulur.

v. Yeni üretilen deneme kromozomların uygunluk değerleri hesaplanır.

vi. Hedef kromozom ve ara kromozom uygunluk değerlerinden iyi olan sonraki nesile katılır.

vii. Algoritmanın sonlandırma kriteri sağlanıyorsa durdurulur, sağlanmıyorsa iii. adımdan devam edilmelir.

Bu işleyişe uygun olarak Şekil 3.13'te DE'nin akış diyagramı verilmiştir.



Şekil 3.13. DE algoritması akış diyagramı

3.3. Parçacık Sürü Optimizasyon Algoritması

3.3.1. Giriş

Parçacık sürü optimizasyon algoritması, Eberhart ve Kennedy tarafından önerilmiş (1995) popülasyon temelli evrimsel bir algoritmadır. Bu algoritma kuş sürülerinin yiyecek arama davranışlarından esinlenerek oluşturulmuştur. Sürü (popülasyon), problemin olası çözümlerini temsil eden kuşlardan (parçacıklar) oluşur. Kuşların yiyecek arama sürecine benzer olarak, optimum veya optimuma yakın çözüm bulabilmek için her bir parçacık kendi pozisyonunu sürüdeki en iyi pozisyona doğru ayarlamaya çalışmaktadır. Bunun için parçacıkların aralarında bilgi paylaşımı yapmaları ve bir önceki tecrübelerinden faydalanmaları gerekmektedir (Kennedy ve Eberhart 1995). PSO'da genetik ve diferansiyel gelişim algoritmalarında bulunan çaprazlama ve mutasyon işlemleri

bulunmamaktadır. Başlangıçta parçacıklar, rastgele olarak yerleştirilmiş ve rastgele hızlara sahip olarak sürüyü oluştururlar. D boyutlu bir problemde n adet parçacığa sahip sürü, Denklem (3.5)'te matris şeklinde ifade edilmektedir.

$$X = \begin{bmatrix} X_{1,1} & \cdots & X_{1,D} \\ \vdots & \ddots & \vdots \\ X_{n,1} & \cdots & X_{n,D} \end{bmatrix} \tag{3.5}$$

Sürü matrisini oluşturan $i.$ parçacık X_i ile ifade edilmektedir. Çözüm uzayında daha çok bölgede arama yapmak için bazı durumlarda tüm sürü en iyi pozisyonunu kullanırken bazı durumlarda ise belirli bir komşuluk tanımı vasıtasıyla yereldeki en iyi pozisyonuna göre hareket etmektedir. Evrimsel süreç boyunca parçacıklar, bu iki en iyi değere göre güncellenerek yeni pozisyonlarını oluşturmaktadırlar. Bunlardan birincisi bir parçacığın o ana kadarki en iyi uygunluk değeridir. Bu değerle ilişkili parçacık algoritmada *lbest* olarak adlandırılmaktadır. Diğeri ise parçacılar tarafından oluşturulan sürüdeki en iyi uygunluk değerine sahip parçacıktır ve bu parçacık *Gbest* olarak isimlendirilir. Bu parçacıkların vektör gösterimi Denklem (3.6) ve Denklem (3.7)'de verilmektedir.

$$lbest_i = [\mathrm{lbest}_{i,1}, \mathrm{lbest}_{i,2}, \dots, \mathrm{lbest}_{i,D}] \tag{3.6}$$

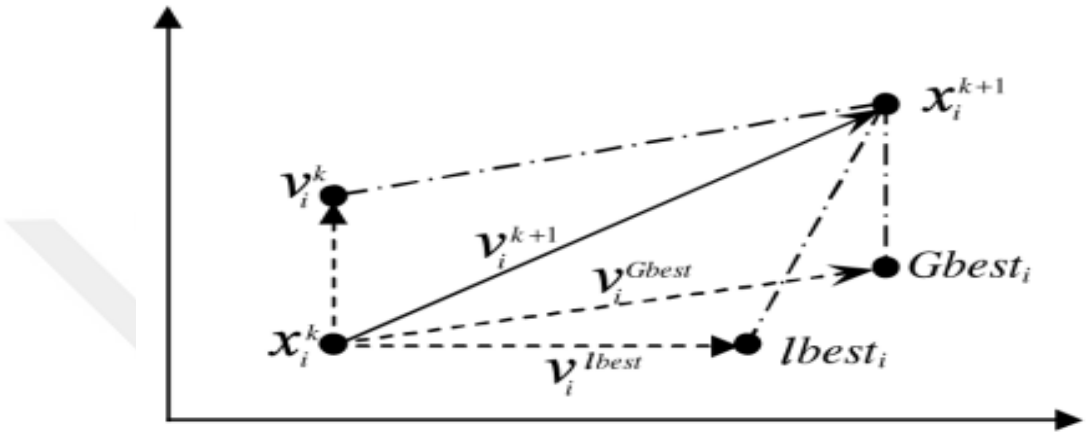
$$Gbest = [\mathrm{l}_1, \mathrm{l}_2, \dots, \mathrm{l}_D] \tag{3.7}$$

O ana kadarki en iyi uygunluk değerini veren i. parçacığın pozisyonu $lbest_i$, tüm parçacıklar arasından en iyi uygunluk değerine sahip ve sürüde tek olan parçacık $Gbest$ olarak ifade edilmektedir. Parçacıklar, çözüm arama sürecinde en iyi çözüme ulaşmak için bu iki değerden yararlanmaktadır (Kennedy ve Eberhart 1995).

3.3.2. Çalışma prensibi

PSO'da çözüm uzayının derinlemesine taranması için evrimsel süreçte parçacıkların farklı pozisyonlarda bulunması beklenmektedir. PSO'da öncelikle parçacıkların en iyi konumları ve sürünün en iyi konumları arasında uzaklıkla ilişkili olarak bir hız vektörü üretilmektedir. Bu hız vektörü parçacığın mevcut pozisyonuna eklenerek parçacığın yeni konumu belirlenmektedir. Şekil 3.14'te parçacığın yeni pozisyon bulma stratejisi

gösterilmektedir. Sürü hafızasında tutulan $Gbest$ ve $lbest$ parçacıkları ve i. parçacığın her konumdaki değişimini ifade eden V_i hız vektörü vasıtasıyla i. parçacık yeni pozisyonunu elde etmektedir.



Şekil 3.14. PSO'da yeni konum bulma stratejisi (Allaoua ve ark. 2009)

Parçacıkların üretilen hız vektörünün büyüklüğüne bağlı olarak çözüm uzayında hızlı veya yavaş hareket etmeleri mümkündür. Parçacıklar çok hızlı hareket ettiği takdirde çözüm uzayının dışına çıkabilmekte, en iyi çözüme yaklaşmak yerine rastgele noktalara gidebilmektedir. Parçacığın çözüm uzayından ayrılma olasılığını azaltmak için PSO'da hız vektörünün maksimum değeri tanımlanmaktadır. Evrimsel süreç boyunca, hız vektörü sürekli güncellenerek olası yeni çözümler üretilmekte ve bu çözümlere dayanarak eğer üretilen çözüm daha iyiyse Gbest ve lbest değerleri güncellenmektedir. PSO'nun kullandığı bazı durdurma kriterleri aşağıda verilmektedir.

PSO,

- Kullanıcı tarafından belirtilen maksimum iterasyon sayısına ulaştığında,
- İstenen minimum değerden daha iyi bir değeri sağladığında,
- İstenilen sonuç birebir elde edildiğinde,
- Belirli iterasyon sayısı boyunca herhangi bir gelişim göstermediğinde,

durdurulabilmektedir.

3.3.2.1. PSO parametreleri

PSO'nun hesaplama yapabilmek için genetik algoritmadaki gibi ikili kodlama yerine reel sayılarla çalışması, mutasyon ve çaprazlama işlemlerine ihtiyaç duymaması, bu algoritmanın kullandığı farklı işlem adımları için farklı parametre tanımlamalarına ihtiyaç doğurmuştur. Genel PSO parametrelerinin listesi ve tanımlamaları aşağıda verilmektedir (Tamer ve Karakuzu 2006).

- Parçacık sayısı (n): Probleme bağlı olarak genellikle 20 ile 40 arasında seçilmektedir. Problemin zorluğuna bağlı olarak ve bazı özel problemler için bu sayı artmaktadır.

- Parçacık boyutu (d): Ele alınacak probleme bağlı olarak değişmektedir.

- Parçacık değer aralığı: Optimize edilecek probleme bağlı olarak değişmekle beraber her bir boyut için değer aralığında tanımlanabilmektedir.

- $Vmax$: Bir parçacığın ulaşabileceği maksimum hız (değişim) değeridir. Genellikle parçacık değer aralığına göre belirlenmektedir. Örneğin bir parçacığın değer aralığı [-20,20] aralığında ise $Vmax$=40 ile sınırlandırılabilir.

- Öğrenme faktörleri (c_1, c_2): Arama uzayında, c_1 parçacığın $lbest$, c_2 ise $Gbest$ değeri etrafında arama yapmasını sağlar. Eberhart ve Shi (2000) yaptığı çalışmada bu katsayıların birbirine eşit olmasını ve 1,494 değerinin alınmasını önermiştir.

- Rastgele sayılar (r_1, r_2): [0 1] aralığında düzgün dağılımlı rasgele sayılardır.

- Eylemsizlik sabiti (w): Parçacığın hızını kontrol etmekte kullanılan katsayıdır. Bu katsayı sabit alınabilmekle beraber, Çizelge 3.2'den görüleceği üzere literatürdeki çalışmalarda dinamik olarakta hesaplanabilmektedir. Bu çalışmada sabit iç ağırlık stratejisi uygulanmıştır.

Çizelge 3.2. İç ağırlık (w) stratejileri

Strateji Adı	Formül	Kaynak
Sabit iç ağırlık	$w = 0{,}7$	(Shi ve Eberhart 1998)
Rastgele iç ağırlık	$w = 0{,}5 + \frac{Rand()}{2}$	(Eberhart ve Shi 2001)
Kaotik rastgele iç ağırlık	$z = 4z(1-z)$ $w = 0{,}5Rand() + 0{,}5z$	(Feng ve ark. 2007)
Kaotik iç ağırlık	$z = 4z(1-z)$ $w = (w_1 - w_2)\frac{iter_{max} - iter}{iter_{max}} + w_2 z$	(Feng ve ark. 2007)
Küresel-yerel en iyi iç ağırlık	$w_i = (1{,}1 - \frac{Gbest_i}{lbest_i})$	(Arumugam ve Rao 2006)

3.3.2.2. Hız ve konum güncelleme

PSO'da parçacıklar evrimsel süreç boyunca arama uzayındaki konumlarını değiştirerek optimal çözüm arayışı içine girmektedirler. Problemin olası çözümlerini temsil eden her bir parçacık, bir hız vektörü hesaplayarak bu hızla yeni konumuna gider ve problemin olası çözümü üretilmiş olur. PSO'da parçacıkların yer değiştirmesi Denklem (3.8)'e göre yapılmaktadır.

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{3.8}$$

Burada, t o anki nesli, X_i o anki parçacığı, V_i parçacığa ait hız vektörünü temsil etmektedir. Parçacığın yeni pozisyonunu belirlemede kullanılan hız vektörünü hesaplamak için Denklem (3.9) kullanılmaktadır. Parçacık, kendisinin ve sürünün o ana kadarki en iyi pozisyonunu kullanarak yeni bir hız formülü oluşturmaktadır.

$$V_i(t+1) = wV_i(t) + c_1 r_1(t)\big(lbest_i(t) - X_i(t)\big) + c_2 r_2(t)\big(Gbest_i(t) - X_i(t)\big) \tag{3.9}$$

Parçacıklar, Denklem (3.8) ve (3.9)'a göre güncellenerek yeni konumlarına ulaşırlar. Her bir parçacık için bu işlem tekrarlanmaktadır. Eylemsizlik sabiti (w) eski hız vektörünün yeni hız vektörü üzerinde etkisini arttırmak veya azaltmak için kullanılmaktadır. Öğrenme

faktörleri (c_1, c_2), yerel ve global arama yeteneği arasındaki dengeyi kurmak amacıyla tanımlanmaktadır.

c_1, parçacığın kendi tecrübelerine göre hareket etmesini sağlarken c_2, sürüdeki en iyi parçacığın tecrübelerine göre hareket edilmesini sağlamaktadır. Bu katsayılardan c_1 katsayısı büyük seçilirse kendi bulduğu en iyi çözüm etrafında (bölgesel, yerel arama), c_2 katsayısı büyük seçilirse küresel en iyi çözüm (genel, global arama) etrafında arama yapma olasılığını arttırmaktadır.

3.3.3. Algoritma adımları

Parçacık sürü optimizasyon algoritmasının genel adımları aşağıdaki şekildedir;

i. Başlangıç sürüsü oluşturulur. Sürüdeki her bir parçacığın başlangıç konumları ve hızları rastgele atanır.

ii. Parçacıkların uygunluk değeri verilen hedef fonksiyonuna göre hesaplanır.

iii. Parçacığın $lbest$ değeri bulunur. Bir önceki adımda hesaplanan parçacık değeri ile parçacığın hafızasında bulunan $lbest$ değeri, uygunluk değerlerine bağlı olarak karşılaştırılır. Eğer bir önceki adımda bulunan parçacığın uygunluk değeri, mevcut $lbest$ parçacığına ait uygunluk değerinden daha iyiyse $lbest$ değeri güncellenir. Diğer durumda aynı kalır.

iv. Parçacığın $Gbest$ değeri bulunur. Adım ii'de hesaplanan sürüdeki en iyi parçacığa ait uygunluk değeri, sürü hafızasında tutulan $Gbest$ değerinden daha iyi ise $Gbest$ değeri güncellenir. Diğer durumda aynı kalır.

v. Parçacıkların arama uzayı içerisinde yeni hız ve konum vektörleri hesaplanır. Bu işlem her bir parçacık için ayrı ayrı yapılır.

vi. Durdurma kriteri sağlanmışsa algoritma sonlandırılır. Bu kriter sağlanmıyorsa adım ii'ye dönülür.

PSO algoritmasının sözde kod yapısı Şekil 3.15'te verilmektedir (Hu 2010).

Başla

Başlangıç parametrelerini belirle.
Başlangıç sürüsünü oluştur.

Durdurma kriteri sağlanıncaya kadar

{

Parçacıkların uygunluk değerini hesapla.
lbest ve *Gbest* değerlerini bul.
Denklem (5.9)'u kullanarak parçacık hızını oluştur.
Denklem (5.8)'i kullanarak parçacıkların konumunu güncelle.

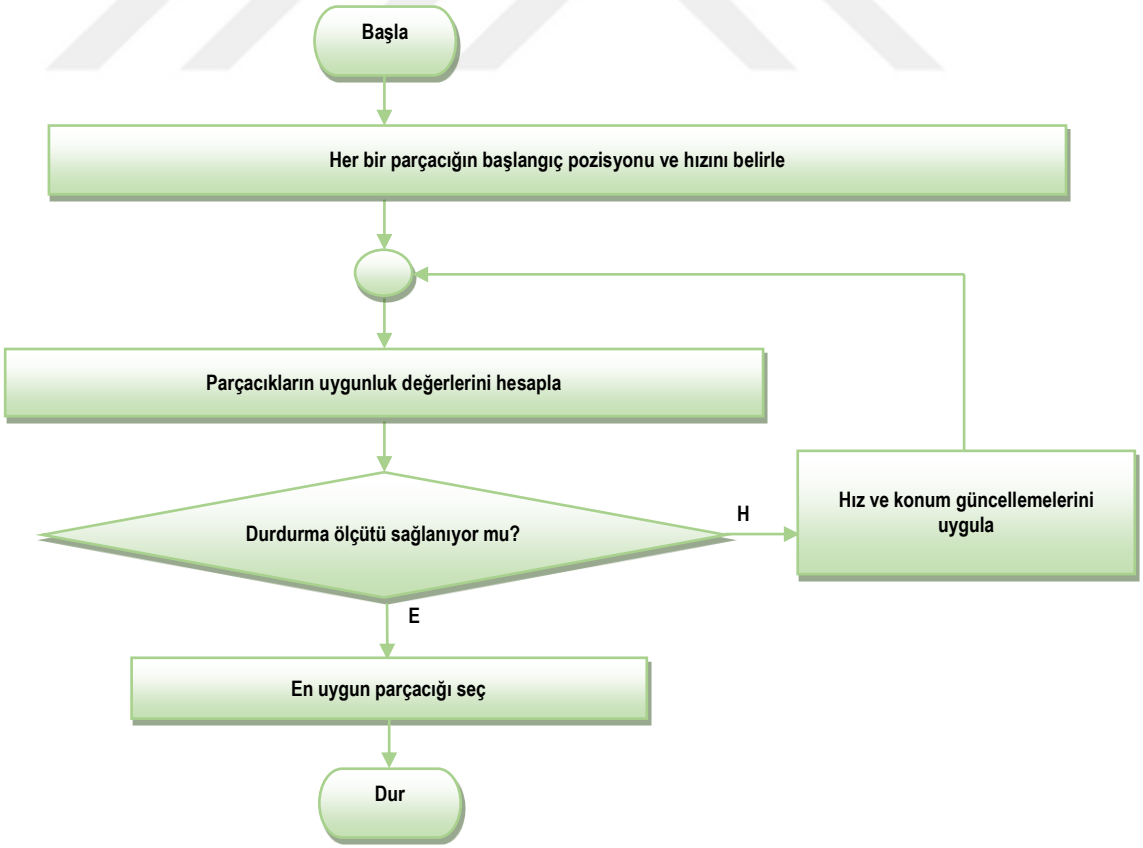
}

En iyi parçacığı çözüm olarak al.

Bitir

Şekil 3.15. Parçacık sürü optimizasyon algoritmasının sözde kodu

Şekil 3.16'da PSO'nun genel akış diyagramı verilmektedir.

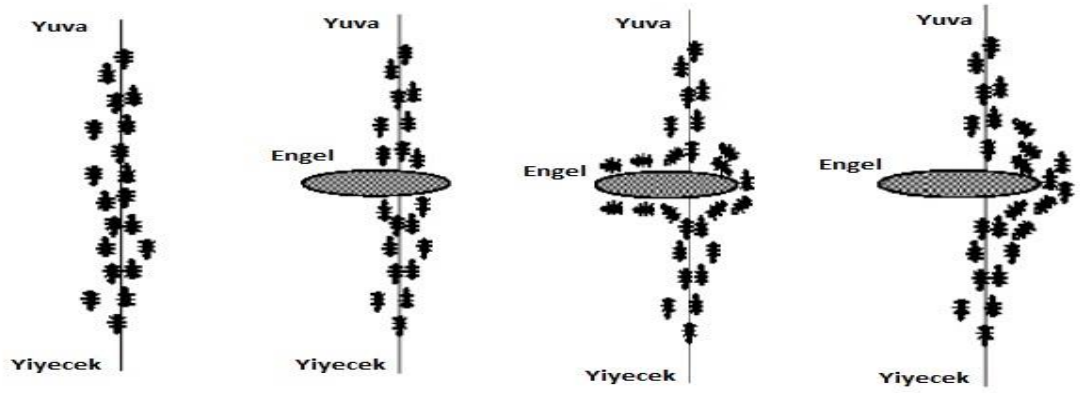


Şekil 3.16. PSO algoritması akış diyagramı

3.4. Karınca Koloni Optimizasyon Algoritması

3.4.1. Giriş

Karınca koloni optimizasyon algoritması, Dorigo tarafından önerilmiş (2001) karınca kolonisi davranışlarından ilham alan ve bu davranışların matematiksel modellemesine dayalı evrimsel bir algoritmadır. Bu algoritma, karıncaların davranışlarını modellemenin yansıra, koloniyi (popülasyon) oluşturan karıncalara (birey) yeni özellikler eklenerek (hafıza, haberleşme) geliştirilmiştir (Karaboğa 2011). Karıncalar yuvaları ile yiyecek arasındaki en kısa yolu bulma kabiliyetine sahip olup, çevresel değişimlere (engel) adapte olabilme yeteneğine sahiptirler. Diğer bir deyişle, karıncalar takip ettikleri yuva ile yiyecek arasında izledikleri en kısa yolda önlerine herhangi bir engel çıkarsa, mevcut şartlarda yeni en kısa yolu bulabilmektedirler. Şekil 3.17'de karıncaların mevcut durumdan, değişen çevre şartlarına bağlı olarak yeni en kısa yolu bulma durumu gösterilmektedir.



Şekil 3.17. Karıncaların en kısa yolu bulma durumu

Karıncalar, yiyecek ile yuva arasındaki en kısa yolu keşfetmek ve aralarında iletişimi sağlamak için feromon maddesini kullanırlar. Hareket halinde bulundukları yola belirli miktarda feromon maddesi bırakarak gideceği yöne karar verebilmektedirler. Bıraktıkları bu maddenin yoğunluğuna göre, yoğun olan yolun tercih edilme olasılığı, daha az yoğun olan tercih edilebilme olasılığına göre daha fazla olmaktadır. Bu prensiple karıncalar en uygun yolu kısa süre içerinde bulabilmektedirler.

3.4.2. Çalışma prensibi

ACO algoritmalarında başlangıçta karıncalar gidecekleri bir sonraki noktayı eşit olasılıklı olarak seçmektedir. Bu seçim her bir yol için eşit olarak atanan feromon izleriyle sağlanır. Daha sonra her bir karıncanın geçiş yapmış olduğu yolların uzunluğuna bağlı olarak feromon miktarları güncellenmektedir. Böylelikle ilgili yoldaki feromon miktar yoğunluğuna bağlı olarak karıncalar en uygun yol tercihi yapabilmektedir. Karıncaların gidebileceği yolları seçme olasılıkları Denklem (3.10)'da formülize edilmektedir (Dorigo 2001).

$$P_k(i,j) = \begin{cases} \frac{\tau(i,j)^{\alpha}\mu(i,j)^{\beta}}{\sum_{z \in j_k(i)} \tau(i,z)^{\alpha}\mu(i,z)^{\beta}} & eğer \quad j \in j_k(i) \\ 0 & Diğer\ durumda \end{cases} \tag{3.10}$$

Denklem (5.10)'da, $\tau(i,j)$ $i.$ ve $j.$ noktalar arasındaki feromon miktarını, $\mu(i,j)$ i ve j. noktaları arasındaki uzaklığın tersi ile ifade edilen sezgisel parametreyi, $j_k(i)$ i noktasındaki karıncanın gidebileceği yolları α ve β ise ayarlanabilir parametreler olarak ifade edilmektedir. α ve β parametrelerinin alacağı farklı değerlere göre, algoritmanın çözümü oluştururken aşağıdaki davranışları izleme eğilimindedir:

- α değerinin yüksek değerler alması durumunda, feromon miktarının fazla olduğu yolların seçilme ihtimali artmaktadır. Bu, karıncaların yollarını seçerken tesadüflük olasılığının azalması demektir. Dezavantajı ise ACO algoritmasının probleme çözüm ararken belirli iterasyondan sonra durgunluk (stagnation) sergileme olasılığının artmasıdır.

- α değerinin düşük olması durumunda, feromon miktarı az olan yolların seçilim ihtimali yükselir. Bu, karıncaların üreteceği çözümlerin durgunluk (stagnation) ihtimalini azaltırken, çeşitlilik olasılığını arttırmaktadır. Dezavantajı ise olası en iyi çözümden uzaklaşarak, karıncaların tesadüfi hareket etme durumunu arttırmasıdır.

- β değeri yol uzunluklarının, bir sonraki noktanın seçimine etkisini belirtmektedir. Bu değer arttıkça yolun seçimindeki tesadüfilik artmaktadır.
- α=0 olması durumu feromon etkisini görmezden gelmek, β =0 olması durumunda ise tamamen feromon etkisine göre hareket etmek anlamına gelmektedir.

Feromon miktarı fazla olan yolun seçilme olasılığı daha yüksektir. $i.$ noktadaki karıncanın gidebileceği $j.$ nokta Denklem (3.11)'deki gibi seçilmektedir.

$$j = max\{\tau(i,z)^{\alpha}\mu(i,z)^{\beta}\}_{z\in j_k(i)} \quad eğer \quad q \leq q_0 \tag{3.11}$$

Denklemde, q_0 olasılığı ile feromon miktarının en yüksek olduğu yol seçilirken, $1-q_0$ olasılığı ile gidilmesi mümkün olan yol tercihleri değerlendirilmektedir. Kolonideki karıncaların tümü seçtikleri yolları tamamladıktan sonra feromon miktarları güncellenmektedir Güncelleme işlemi için tüm yollarda feromonlar belirlenen oranlarda (buharlaşma oranı) sırasıyla buharlaştırılmaktadır. Buharlaştırma oranı 0 ve 1 arasında sabit bir değere sahiptir. Karıncaların turlamış oldukları yollardaki feromon miktarları, o yolu önceden kullanan karıncanın toplam yol uzunluğuyla ters orantılı olarak artmaktadır. Bu, kısa yollardan geçmiş olan karıncaların feromon miktarlarında daha fazla artışa sebep olup iyi çözümlerin öneminin arttırılmasını sağlayacaktır (Dorigo ve Gambardella 1997). Karınca koloni algoritmasında, $t+1$ anındaki feromon güncellemesi Denklem (3.12)'de verilmektedir (Karaboğa 2011).

$$\tau(i,j)_{t+1} = p\,\tau(i,j)_t + \Delta\tau(i,j)_{(t,t+1)} \tag{3.12}$$

$\tau(i,j)_t$ t anındaki $i.$ ve $j.$ noktalar arasındaki feromon miktarını, p buharlaşma katsayısını belirtmektedir. Birim zamanda $i.$ ve $j.$ noktalar arasına bırakılan feromon miktarı Denklem (3.13)'teki gibi ifade edilmektedir.

$$\Delta\tau(i,j)_{(t,t+1)} = \sum_{k=1}^{n} \Delta\tau(i,j)_{(t,t+1)}{}^{k} \tag{3.13}$$

Yukarıdaki denklemde, $\Delta\tau(i,j)_{(t,t+1)}{}^{k}$ t ve t+1 zaman aralığında $k.$ karınca tarafından $i.$ ve $j.$ noktalar arasında bırakılan feromon maddesi miktarını temsil etmektedir. Kolonideki

karıncalar güncellenen feromon miktarlarına bağlı olarak turlarını değiştirerek en uygun çözüm arayışını sürdürmektedirler.

3.4.3. Algoritma adımları

Karınca kolonisi optimizasyon algoritmasının sözde kod yapısı Şekil 3.18'deki gibi verilmektedir (Karaboğa 2011).

Başla
Başlangıç parametrelerini belirle.
Başlangıç kolonisini rastsal olarak oluştur.
Durdurma kriteri sağlanıncaya kadar
{
Her bir karıncanın gideceği yolların uzunluklarını hesapla.
Feromon güncellemesi yap.
Güncellenen feromon miktarına bağlı olarak yeni yollar üret.
}
En kısa yola sahip karıncayı çözüm olarak al.
Bitir

Şekil 3.18. Karınca koloni optimizasyon algoritmasının sözde kodu

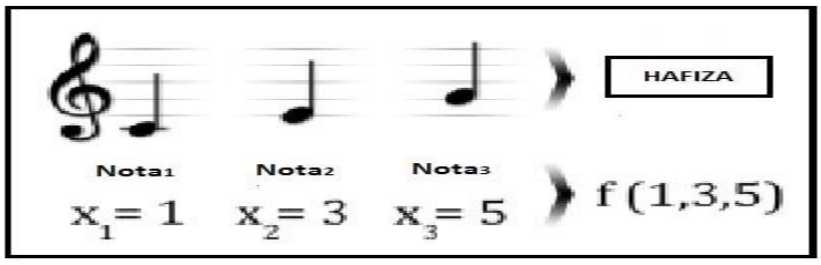
Şekil 3.19'da ACO'nun genel akış diyagramı verilmektedir.

Şekil 3.19. ACO algoritması akış diyagramı

3.5. Armoni Arama Algoritması

3.5.1. Giriş

Bir optimizasyon algoritması olarak kullanılan ve ilk olarak Geem ve arkadaşları tarafından (2001) geliştirilen HS algoritması, müzik tabanlı olup, bir orkestradaki müzisyenlerin çaldıkları notalar vasıtasıyla çıkardıkları müziğin matematiksel olarak modellenmesi prensibine dayanmaktadır. Gerçek hayatta, müzisyenler beste yaparken kullandıkları enstrümanlar vasıtasıyla farklı armoniler elde ederek kulağa en hoş gelen müziği elde etmeye çalışmaktadırlar. Bu armoniler içinde, tecrübelerine göre beğenmediklerini veya uygun olmayanları elerlerken, iyi olanları daha iyi armoniler elde etmek için kullanabilmektedirler. Bu esnada kullanılan enstrümanların akort ayarları tekrar yapılabilmekte, kullanılan notalar değiştirilerek mevcut armoniden daha iyi bir armoni elde edilmeye çalışılabilmektedir. Müzisyenlerin yeni armoniler üretmesindeki bu süreç ile HS algoritmasının kullandığı yöntemler arasındaki benzerlikler şu şekilde açıklanabilir; her bir müzik enstrümanı çözümün bütününü oluşturan (çözüm vektörü) notalarını meydana getirmektedir. Her bir nota, çözüm vektörü üzerindeki bir değişkeni tanımlamaktadır. Şekil 3.20'de bu benzerlik şematize edilmektedir. Bir orkestradaki müzisyenlerin deneyimlerini kullanarak daha önceden bildikleri, uyumlu notaları veya rastgele notalar çalarak, bu notaların bir kombinasyonunu oluşturarak en iyi armoniye ulaşmaya çalışmaları gibi, HS algoritması da rastgele başlatılan notaları kullanarak, önceden denenmiş ve kaydedilmiş olası çözümlerden yola çıkar ve optimum çözüme ulaşmaya çalışır (Lee ve Geem 2004).



Şekil 3.20. HS algoritması ve müzik enstrümanı arasındaki benzetim

3.5.2. Çalışma prensibi

HS algoritması çözümü elde ederken temel olarak beş adım kullanmaktadır; problemin ve çözüm parametrelerinin belirlenmesi, armoni belleğinin oluşturulması, yeni armoni üretilmesi, armoni belleğinin güncellenmesi, durdurma kriteri kontrolü (Geem 2001, Lee ve Geem 2004). Bu algoritmanın başlıca avantajları arasında karar değişkenleri için özel bir başlangıç çözümü tanımlanmasına gerek duyulmaması, arama uzayında en iyi çözümün birden çok yönde aranması ve bu sayede yerel minimum noktalarından kurtulabilmesi, optimizasyon işleminde sürekli değişkenler kullanılabildiği gibi genetik algoritmaya benzer olarak sıfır-bir değeri alan değişkenlerin de kullanılabilmesi olarak söylenebilmektedir (Lee ve Geem 2004, Yang 2010). HS algoritması kullanılırken bir problemin çözümünde öncelikle Denklem (3.14)'te verildiği gibi ilgili problemin matematiksel modellemesi yapılır.

$$F = \min\{f(x)\} \quad x_i \in X_i = 1,2,\dots M \tag{3.14}$$

Burada $f(x)$ minimize edilecek hedef fonksiyonunu, x_i problemin değişkenini, X_i her bir değişken için kullanılacak çözüm uzayını, M ise toplam değişken sayısını ifade etmektedir. HS algoritmasında başlangıçta belirlenmesi gereken bazı parametreler mevcuttur. Bunlar; armoni belleği kapasitesi (HMS), armoni belleği dikkate alma oranı ($HMCR$) ve ton ayarlama oranıdır (PAR). Çözüm uzayı içerisinde, D boyutlu bir problemin çözümü için Denklem (3.15)'tekine benzer olarak, rassal olası çözümleri içeren bir armoni belleği oluşturulur.

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} \\ \vdots & \ddots & \vdots \\ x_{HMS,1} & \cdots & x_{HMS,D} \end{bmatrix} \tag{3.15}$$

Bir karar değişkeni, mevcut armoni belleğinden 0 ile 1 arasında değişen $HMCR$ olasılıkla seçilebilirken, $(1 - HMCR)$ olasılıkla mevcut çözüm uzayı içerisinden rassal olarak seçilebilmektedir. Seçilim işlemi Denklem (3.16)'da verilmektedir.

$$x_i' = \begin{cases} x_i' \in \{x_{1,i}, x_{2,i}, \dots . x_{HMS,i}\} & HMCR\ olasılığında \\ x_i' \in X_i & Diğer\ durum \end{cases} \tag{3.16}$$

Yeni armoni vektörü $x' = (x'_1, x'_2, \ldots x'_D)$ armoni belleğinde bulunan tonlardan rassal olarak seçilmektedir. Değişkenlerin armoni belleğinden seçilimi 0 ile 1 arasında belirlenen $HMCR$ olasılığına göre olmaktadır. Bu işlemden sonra PAR parametresine bağlı olarak ton ayarlama işleminin gerekli olup olmadığı Denklem (3.17)'ye göre belirlenmektedir.

$$x'_i = \begin{cases} x'_i \pm Rand(0,1) * bw & PAR\ olasılığında \\ x'_i & Diğer\ durum \end{cases} \quad (3.17)$$

Denklem (3.17)'de, bw parametresi bant genişliğini, $Rand(0,1)$ 0 ile 1 arasında üretilmiş rastgele sayıyı ifade etmektedir. Eşitlikten görüleceği üzere $1 - PAR$ olasılığında x'_i karar değişkeni aynı kalmaktadır. Bu adımdan sonra oluşturulan yeni armoni vektörü ile hedef fonksiyonu sonuçlarına göre, armoni hafızasında bulunan en kötü armoni vektörü arasında bir seçilim işlemi uygulanmaktadır. Hedef fonksiyon değerlerine göre, yeni oluşturulan armoni vektörü, en kötü armoniden iyiyse en kötü armoni vektörü hafızadan çıkarılır ve yeni armoni vektörü onun yerine yerleştirilir.

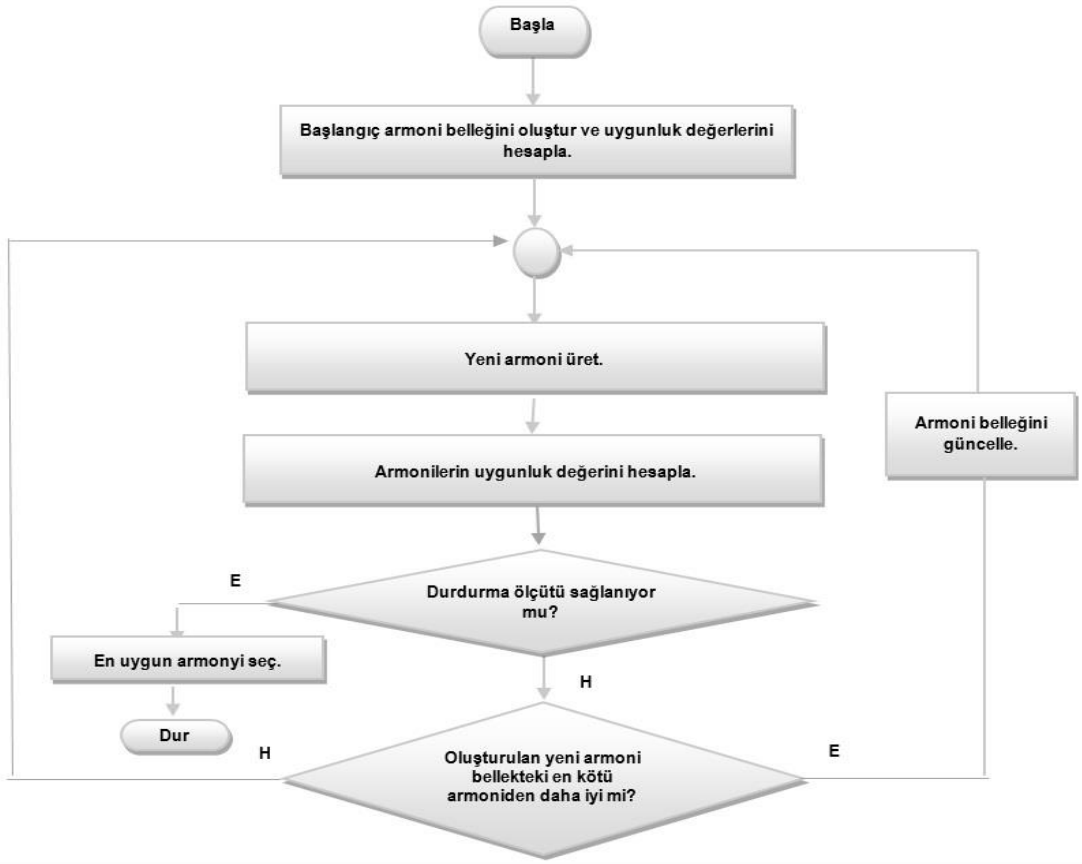
3.5.3. Algoritma adımları

HS algoritmasının sözde kod yapısı aşağıdaki gibi verilmektedir (Geem 2001).

Başla
Başlangıç parametrelerini belirle.
Başlangıç armoni hafızasını rastsal olarak oluştur.
Hafızadaki armonilerin uygunluk değerinlerini hesapla.
Durdurma kriteri sağlanıncaya kadar
{
Denklem (3.16-17)'yi kullanarak armonileri güncelle.
Güncellenen armonilerin uygunluk değerilerini hesapla.
Eğer yeni armoni hafızadaki en kötü armoniden daha iyiyse yer değiştir, değilse mevcut düzeni koru.
}
En iyi uygunluk değerine sahip armoniyi çözüm olarak al.

Şekil 3.21. Armoni arama algoritmasının sözde kodu

Şekil 3.22'de HS algoritmasının genel akış diyagramı verilmektedir.



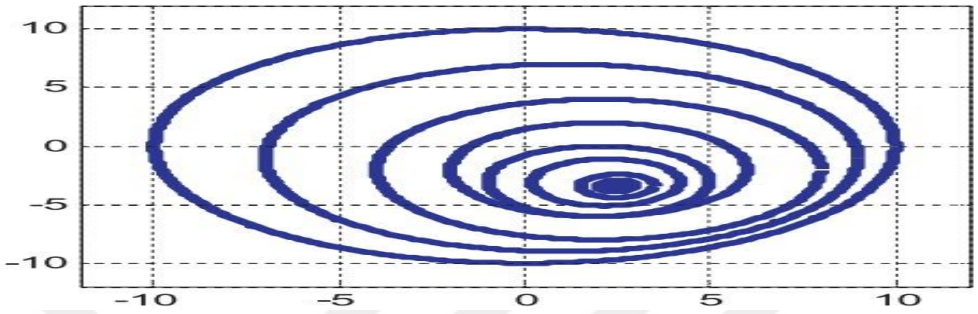
Şekil 3.22. HS algoritması akış diyagramı

3.6. Girdap Arama Algoritması

3.6.1. Giriş

Girdap arama algoritması, Doğan ve Ölmez tarafından önerilmiş (2015), sıvı yüzeyinde akışkanların oluşturduğu girdap şeklinden ilham alınarak geliştirilen bir evrimsel bir algoritmadır. Bu algoritma adaptif arama yeteneği sayesinde, yeni çözümler keşfetmenin (exploration) yanı sıra komşu çözümler arasında gösterdiği yerel araştırma mekanızmasından yararlanarak (explotation) optimal çözümüne ulaşmaktadır. Bu sebeple algoritma çözüm uzayında başlangıç adımlarında küresel bir arama yaparak olası çözümleri ararken, ileriki adımlarda bulduğu olası çözümleri geliştirerek optimal çözüme

yakınsamaktadır. VS algoritmasının, [-10 10] aralığındaki 2 boyutlu optimizasyon problemi için temsili girdap yapısı Şekil 3.23'te verilmektedir (Doğan ve Ölmez 2015).



Şekil 3.23. VS algoritması temsili girdap yapısı (Doğan ve Ölmez 2015)

3.6.2. Çalışma prensibi

VS algoritmasını 2 boyutlu bir optimizasyon problemi üzerinden yola çıkarak, problem iç içe geçmiş çemberler şeklinde modellenebilir. En dıştaki, en geniş çemberin arama uzayının sınırlarını belirlediğini düşünürsek çemberin merkezi Denklem (3.18) kullanılarak hesaplanabilir.

$$\mu_0 = \frac{\text{üst}sınır + altsınır}{2} \tag{3.18}$$

üst$sınır$ ve $altsınır$ problemin üst ve alt limitlerini, μ_0 ise en dış girdap çemberinin merkezini temsil etmektedir. Başlangıçta rassal olarak Gauss dağılımı kullanarak μ_0 merkezi etrafında $C_t(s)$ komşu çözümleri üretilir. t iterasyon indeksinde, $t = 0$ başlangıç konumunda $C_0(s) = \{s_1, s_2 \dots s_n\}$ $n = 1,2,..M$ başlangıç olası çözüm kümesi elde edilir. M toplam olası çözüm sayısı olarak ifade edilmektedir. μ_0 merkezli Gauss dağılımı ile üretilen olası çözümler için belirlenen standart sapma σ_0, Denklem (3.19)'da verilmektedir.

$$\sigma_0 = \frac{\max(\text{üst}sınır) + \min(altsınır)}{2} \tag{3.19}$$

Başlangıçta σ_0 değeri problemin üst ve alt limitlerin maksimum ve minimum değerlerini kullanarak belirlenmektedir. Daha sonra, $C_0(s)$ olası çözüm kümesi içerisinden bir çözüm seçilmektedir. Bu çözüm hedef fonksiyon uygunluk değerine bağlı olarak en iyi çözüm

olarak kabul edilmektedir. Üretilen olası çözüm kümesinin elemanları, problemin alt ve üst limitlerinin dışına çıkması durumunda Denklem (3.20) ve (3.21) kullanılarak tekrar limitler içine alınmaktadır.

$$s_n^i = rand.\left(\text{üst}sınır^i - altsınır^i\right) + altsınır^i\ ,\ s_n^i < altsınır^i \qquad (3.20)$$

$$s_n^i = rand.\left(\text{üst}sınır^i - altsınır^i\right) + altsınır^i\ ,\ s_n^i > \text{üst}sınır^i \qquad (3.21)$$

Denklem (3.20) ve (3.21)'de $i = 1,2,..D$ olmak üzere D problemin boyutunu ifade etmektedir. O anki çözüm, çözümü geliştirme süreci boyunca bulunan en iyi çözümden uygunluk değerine bağlı olarak daha iyiyse s_{best} değeri olarak atanır. s_{best} çözümü bir sonraki adımda olası çözümlerin üretilmesinde kullanılacak olan merkez olarak belirlenir. Gauss dağılımıyla yeni çözüm arama işlemi durdurma kriteriyle algoritma sonlandırılıncaya kadar devam etmekte ve algoritma bu kriter sağlandığında s_{best} çözümü en iyi çözüm olarak alınmaktadır.

3.7. Geri İzleme Arama Optimizasyon Algoritması

3.7.1. Giriş

Geri izleme arama optimizasyon algoritması popülasyon tabanlı, iteratif bir algoritmasıdır. Temel olarak 5 ana işlem adımından oluşmaktadır. Bunlar; başlangıç, seleksiyon-1, mutasyon, çaprazlama ve seleksiyon-2 olmaktadır. BS algoritmasının genel yapısı Şekil 3.24'te verilmektedir (Civicioglu 2013).

1-Başlangıç
Tekrarla
2- Seleksiyon-1
Geçici popülasyonun oluşturulması
3- Mutasyon
4- Çaprazlama
Dur
5- Seleksiyon-2
Durdurma kriteri

Şekil 3.24. BS'nin genel yapısı

3.7.2. Çalışma prensibi

BS algoritmasında başlangıç popülasyonu Denklem (3.22) kullanılarak oluşturulur.

$$Pop_{i,j} \sim U\left(altsınır_j, \text{üst}sınır_j\right) \quad i = 1,2, \dots N \; ve \; j = 1,2 \dots D \qquad (3.22)$$

$altsınır$ ve üst$sınır$ problemimizin alt ve üst sınırlarını, N ve D sırasıyla popülasyon sayısı ve problemimizin boyutunu ifade etmektedir. Seleksiyon-1 işleminde, başlangıç esnasında eski popülasyon ($oldPop$) Denklem (3.23)'teki gibi hesaplanır. Bu popülasyon BS algoritmasında arama doğrultusunu hesaplamak için kullanılmaktadır.

$$if \; a < b, oldPop \coloneqq Pop|a, b \sim U(0,1) \qquad (3.23)$$

Denklem (3.23)'te, $\coloneqq$ güncelleme operatörüdür. BS algoritması daha önceden rassal olarak gelen popülasyonu, eski popülasyon ($oldPop$) olarak kullanır ve bu eski popülasyon ileride değiştirilmek üzere algoritmanın hafızasında tutulur. Denklem (3.24) kullanılarak, $oldPop$ popülasyonunun elemanları rassal olarak değiştirilir.

$$oldPop \coloneqq permuting(oldPop) \qquad (3.24)$$

Denklem (3.24)'te $permuting$ fonksiyonu karıştırma fonksiyonu olarak adlandırılmaktadır. Daha sonra mutasyon işlemi Denklem (3.25) kullanılarak gerçekleştirilir.

$$Mutant = P + F.(oldP - P) \qquad (3.25)$$

Denklem (3.25)'te F, $(oldP - P)$ arama doğrultu matrisinin büyüklüğünü kontrol etmektedir. Geçmiş popülasyon $oldP$ kullanılarak arama doğrultu matrisi hesaplanmakta, böylelikle kısmi olarak önceki nesildeki bilgiler işleme dahil edilmektedir. Çaprazlama işlemininn, deneme popülasyonunun (T_{pop}) başlangıç değeri $Mutant$ olarak ayarlanmaktadır. BS'nin mutasyon işlemi iki aşamadan oluşmaktadır. İlk aşamada $N.D$ büyüklüğünde ikili sayı sisteminde tam değerli bir matris üretilmekte (map) ve bu matris vasıtasıyla T_{pop} popülasyonunu elemanları üzerinde değişiklik yapılıp yapılmayacağına

karar verilmektedir. Algoritma, $map_{i,j} = 1$ olması durumunda T_{pop} popülasyonu üzerinde güncelleme yapmaktadır. Seleksiyon-2 aşamasına geçildiğinde, P_i ve T_i değerleri , P_i'leri güncellemek amacıyla kullanılmaktadır. Eğer P'nin en iyi değeri P_{best}, algoritmanın o ana kadarki bulduğu global minimum değerinden daha iyiyse, en iyi çözüm değeri P_{best} olarak güncellenmektedir (Civicioglu 2013).

4. EVRİMSEL ALGORİTMALARLA FİLTRE TASARIMLARI

4.1. Analog Filtre Tasarımı

Bu bölümde, DE, GA, PSO, ACO, HS, VS ve BS algoritmalarıyla literatürdeki analog filtre tasarımlarında sıklıkla kullanılan SV ve Sallen-Key filtre topolojileri kullanılarak alçak geçiren filtreler tasarlanacaktır. Algoritmalar 10 bağımsız çalıştırılmaya tabi tutulmakta, bu çalıştırılmalarda maksimum iterasyon sayısı 10000 ve popülasyon sayısı 50 olarak belirlenmektedir. Filtrelerde kullanılacak devrelerin gerçeklenmesini kolaylaştırmak açısından, algoritmaların bulacağı devre elemanları standart endüstride üretilmiş serilerden seçilecektir. Tasarım örneklerinde, her bir algoritmanın bulduğu hatalar ve yakınsama grafiklerine bağlı olarak, tasarlanan filtre türü üzerindeki üstünlükleri ve zayıflıkları ele alınacaktır. Her bir tasarım örneğimiz için, algoritmalar tarafından tasarlanan filtrenin devre elemanlarına bağlı olarak PSpice programı kullanılarak tasarımların doğrulukları gözlemlenecektir.

4.1.1. Durum değişkenli filtre topolojisi kullanarak analog filtre tasarımı

Bu uygulamada, SV filtre topolojisi kullanılarak, literatürdeki çalışmalardan yola çıkarak (aynı hata fonksiyonu ve üretim serilerini kullanarak) ikinci dereceden alçak geçiren analog filtre tasarımı gerçekleştirilecektir. Analog filtre tasarımında kullanacağımız devre elemanları literatürdeki çalışmalara uygun olarak E24 endüstri serilerinden seçilmektedir. Örneğimizin tasarımı, PSpice benzetim programı aracılığıyla yapılıp, algoritmalarımızın tasarım sonuçları, bu program aracılığıyla doğrulanmaktadır.

4.1.1.1. Hedef fonksiyon

SV ikinci dereceden alçak geçiren filtre devresi Şekil 4.1'de görülmektedir. Filtremizin devre elemanı değerlerine bağlı olarak kesim frekansı ve kalite faktörü Denklem (4.1) ve Denklem (4.2)'de verilmektedir.

$$\omega_{SV} = \sqrt{\frac{R_4}{R_3}\left(\frac{1}{C_1 C_2 R_5 R_6}\right)} \tag{4.1}$$

$$Q_{SV} = \frac{R_3(R_1+R_2)}{R_1(R_3+R_4)}\sqrt{\frac{C_1R_4R_5}{C_2R_3R_6}} \quad (4.2)$$

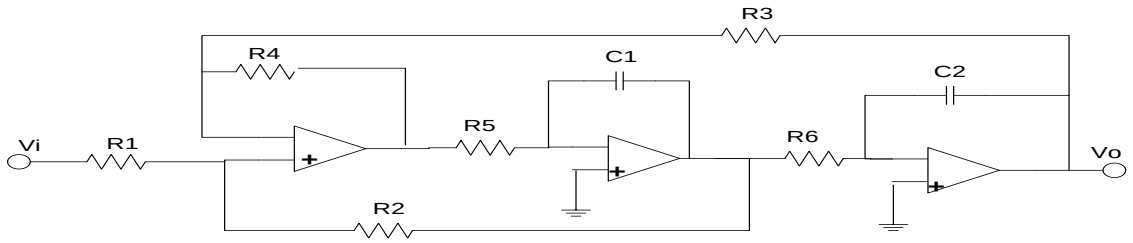
Hedef kesim frekansı (ω_c) 10 krad/sn, hedef kalite faktörü (Q_h) 0,707 olarak alınmıştır. Toplam tasarım hatası (J) Denklem (4.3)'te tanımlanmaktadır. Bu hata, Denklem (4.4)'te verilen kesim frekansı sapması (Δω) ve Denklem (4.5)'de verilen kalite faktörü sapmasının (ΔQ), 0.5 ile çarpımlarının toplamı şeklinde ifade edilebilmektedir (Bishnu ve ark. 2015).

$$J = 0{,}5(\Delta\omega + \Delta Q) \quad (4.3)$$

$$\Delta\omega = \frac{|\omega_{SV}-\omega_c|}{\omega_c} \quad (4.4)$$

$$\Delta Q = \frac{|Q_{SV}-Q_h|}{Q_h} \quad (4.5)$$

Tasarımda kullanılacak ayrık elemanlardan, direnç değerleri 10^3 ile 10^6 Ω aralığında ve kapasite değerleri 10^{-9} ile 10^{-6} F aralığında, E24 devre elemanı üretim serilerine uygun olarak seçilmektedir. J hata fonksiyonumuz, kullanılan algoritmalar tarafından minimize edilerek, ilgili devre elemanı değerlerine ulaşılması amaçlanmaktadır.



Şekil 4.1. Durum değişkenli ikinci dereceden alçak geçiren filtre devresi

4.1.1.2. Örnek tasarım ve analizi

Bu örnek uygulamada, Şekil 4.1'de verilen devreki devre elemanı dizilimine uygun olarak ikinci dereceden alçak geçiren filtreyi SV topoloji kullanarak DE, GA, PSO, ACO, HS, VS ve BS algoritmaları ile tasarımı gerçeklenmiştir. Alçak geçiren analog filtrenin algoritmalar tarafından bulunan devre elemanları Çizelge 4.1'de verilmektedir. 10 bağımsız çalıştırma sonucu bulunan hatalar Çizelge 4.2'de, bağımsız çalıştırmalar sonucu

bulunan en iyi hatanın algoritmalara ait yakınsama grafiği Şekil 4.2'de gösterilmektedir. Algoritmalar tarafından bulunan devre elemanı değerlerine uygun olarak, her bir algoritma için PSpice programı aracılığıyla besleme gerilimi ±12 V verilen LM741 op-amp kullanılarak SV ikinci dereceden alçak geçiren filtrenin benzetim grafiği Şekil 4.3'te verilmektedir.

Çizelge 4.1. Durum değişkenli ikinci dereceden alçak geçiren filtre için algoritmaların bulduğu devre elemanları

Devre Elemanları (E24)	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
R_1(kΩ)	430	24	680	33	470	16	150
R_2(kΩ)	1,5	5,6	20	5,1	36	68	9,1
R_3(kΩ)	5,6	18	16	4,7	18	6,2	10
R_4(kΩ)	13	91	10	100	390	270	20
R_5(kΩ)	18	6,2	18	62	62	20	100
R_6(kΩ)	16	16	56	12	16	33	1
C_1(nF)	13	56	6,2	22	24	30	2
C_2(nF)	6,2	9,1	1	13	9,1	22	100

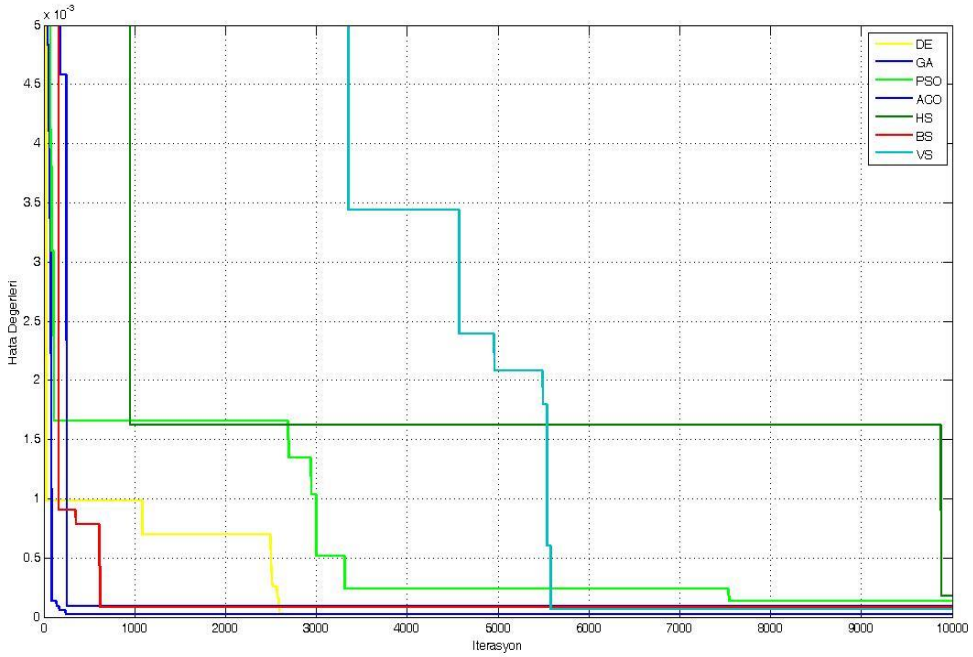
Çizelge 4.2'den görüldüğü üzere, DE algoritması 10 bağımsız çalıştırma için GA algoritmasından küçük bir farkla daha iyi bir hata değerine ulaşmasına rağmen, bağımsız çalıştırma sayısına bağlı olarak GA algoritması en iyi ortalama hata değerine sahip olduğundan dolayı bu örnek tasarım problemi üzerinde daha tercih edilebilir bir algoritma olduğu söylenebilmektedir. Ortalama hata değerleri analiz edildiğinde, ACO ve HS algoritmalarının diğer algoritmalara nazaran iyi sonuçlara ulaşamadığı görülmektedir. Bu, ACO ve HS algoritmalarının bu problemimiz üzerinde kararlı tasarımlar sunamadığını göstermektedir. Diğer yandan, aynı tabloya bağlı kalarak, HS algoritması en iyi bulunan hata bakımından kıyaslamaya tabi tutulan algoritmalar arasından en kötü performansı gösterdiğinden ve ortalama hata değeri bakımından iyi bir sonuca ulaşamadığından dolayı bu tasarım örneğinde tercih bakımından iyi bir seçenek olarak göze çarpmamaktır. VS ve

BS algoritmaları bulmuş oldukları minimum en iyi hata ve ortalama hata değerleri açısından birbirlerine yakın sonuçlar elde etmektedir.

Çizelge 4.2. Durum değişkenli ikinci dereceden alçak geçiren filtre için algoritmalar tarafından 10 bağımsız çalıştırmada bulunan hata değerleri

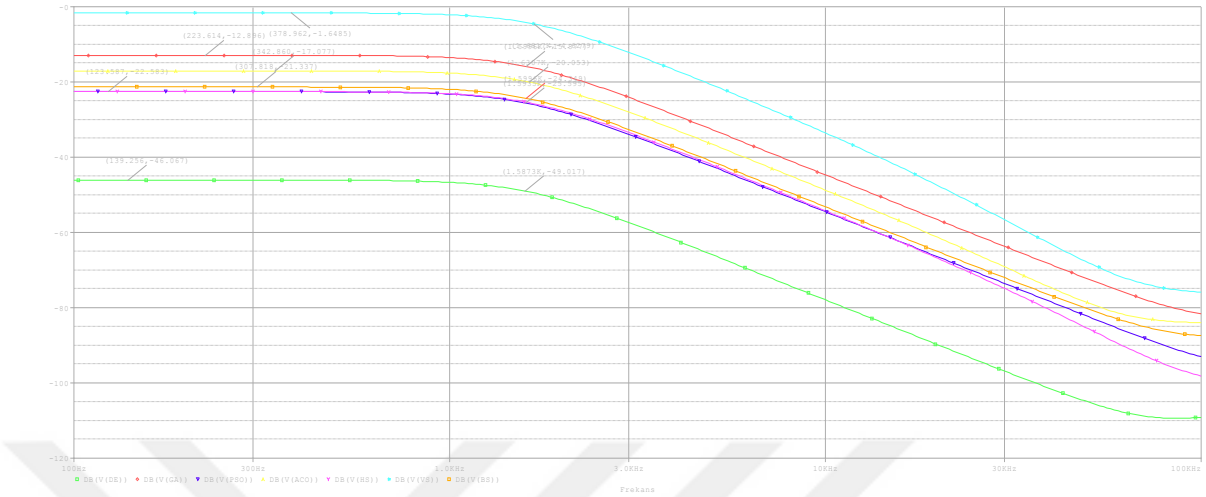
Hata değerleri	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
Minimum en iyi hata	$1{,}857\times10^{-5}$	$2{,}240\times10^{-5}$	$1{,}319\times10^{-4}$	$9{,}170\times10^{-5}$	$1{,}743\times10^{-4}$	$6{,}810\times10^{-5}$	$7{,}857\times10^{-5}$
Ortalama hata	$5{,}852\times10^{-4}$	$6{,}639\times10^{-5}$	$3{,}477\times10^{-4}$	0.002	0.0012	$4{,}452\times10^{-4}$	$5{,}924\times10^{-4}$
Maksimum en iyi hata	0,0018	0,0016	0,0010	0.0054	0.0029	$9{,}418\times10^{-4}$	0,0011

Şekil 4.2'den görüleceği üzere, GA algoritması kendi en iyi çözümüne en hızlı (yaklaşık 200 iterasyon) yakınsayan algoritma olmaktadır. Bu algoritma, kendi en iyi çözümüne, algoritmalar arasındaki en iyi çözüme sahip DE algoritmasından az bir farkla daha kötü olarak elde etmektedir. Diğer yandan DE algoritması yaklaşık 500. iterasyondan sonra kendi en iyi çözümüne yakınsadığından dolayı, hem yakınsama hızı hem hata göz önünde bulundurulduğunda DE algoritmasının bu filtre tasarımı için uygun olduğu görülebilmektedir. HS algoritması yakınsadığı hata değerinin en yüksek ve yakınsama hızının en düşük olması sebebiyle algoritmalar arasında en kötü performansı sergilemektedir. Şekil 4.3'te verilen PSpice benzetim grafiğinde, her bir algoritmanın başlangıç kazanç değerleri ve bu kazanç değerlerinin 3 dB düştüğü kesim frekansı noktaları işaretlenmiştir. PSpice benzetimi aracılığıyla devrenin kesim frekanslarını: GA ve BS algoritmaları; 10,05 krad/sn, PSO ve HS algoritmaları; 10,28 krad/sn, DE algoritması; 10,01 krad/sn, VS algoritması; 10,56 krad/sn ve ACO algoritması; 10,09 krad/sn bulmaktadır. Bu sonuçlar, algoritmaların ön tanımlı olarak verdiğimiz 10 krad/sn kesim frekansına iyi bir yakınsama gösterdiğini doğrulamaktadır. Diğer yandan algoritmalar vasıtasıyla tasarlanan filtreye ilişkin yüksek doğruluğa bağlı olarak, geçirme bandında herhangi bir dalgalanma gözlemlenmemektedir.



Şekil 4.2. Durum değişkenli ikinci dereceden alçak geçiren filtre için algoritmaların en iyi hata değerine yakınsama grafiği

Şekil 4.3'te verilen PSpice benzetim grafiğinde, her bir algoritmanın başlangıç kazanç değerleri ve bu kazanç değerlerinin 3 dB düştüğü kesim frekansı noktaları işaretlenmiştir. PSpice benzetimi aracılığıyla devrenin kesim frekanslarını: GA ve BS algoritmaları; 10,05 krad/sn, PSO ve HS algoritmaları; 10,28 krad/sn, DE algoritması; 10,01 krad/sn, VS algoritması; 10,56 krad/sn ve ACO algoritması; 10,09 krad/sn bulmaktadır. Bu sonuçlar, algoritmaların ön tanımlı olarak verdiğimiz 10 krad/sn kesim frekansına iyi bir yakınsama gösterdiğini doğrulamaktadır. Diğer yandan algoritmalar vasıtasıyla tasarlanan filtreye ilişkin yüksek doğruluğa bağlı olarak, geçirme bandında herhangi bir dalgalanma gözlemlenmemektedir.



Şekil 4.3. Durum değişkenli ikinci dereceden alçak geçiren filtre için algoritmaların PSpice benzetim grafiği

4.1.2. Sallen-Key filtre topolojisi kullanarak analog filtre tasarımı

Bu bölümde Sallen-Key filtre topolojisi kullanılarak, literatürde sıklıkla kullanılan dördüncü dereceden Butterworth alçak geçiren analog filtre tasarımı gerçekleştirilecektir. Tasarım kriterleri ve tasarım parametreleri algoritmaya tanıtılarak, hedeflenen kısıtlamalar dahilinde en az tasarım hatasına sahip filtreyi oluşturan pasif eleman değerlerinin algoritmalar aracılığıyla saptanması hedeflenmektedir. Analog filtre tasarımında kullanılacak olan devre elemanları E12 endüstri serilerinden seçilmektedir. Filtre devremiz, PSpice simülasyon programı aracılığıyla simüle edilip, bulunan devre elemanlarının doğruluğu gözlemlenecektir.

4.1.2.1. Hedef fonksiyon

Butterworth filtreler işareti geçirme bandında en az bozunumla ve tercih edilebilir bir zayıflatma oranında filtrelemek için kullanılabilecek filtrelerdendir. Dördüncü dereceden Butterworh alçak geçiren filtre, iki adet ikinci dereceden bloğun ardışık bağlanmasıyla elde edilebilmektedir. Dördüncü dereceden Butterworth alçak geçiren filtre devresi Şekil 4.4'te görülmektedir. Ardışık filtre yapısını oluşturan her bir ikinci dereceden Butterworth alçak geçiren filtrenin devre elemanı değerlerine ilişkin kesim frekansı denklemleri Denklem

(4.6) ve Denklem (4.7)'de kalite faktörü denklemleri Denklem (4.8) ve (4.9)'da verilmektedir.

$$\omega_{SK1} = \frac{1}{\sqrt{R_1 R_2 C_1 C_2}} \tag{4.6}$$

$$\omega_{SK2} = \frac{1}{\sqrt{R_3 R_4 C_3 C_4}} \tag{4.7}$$

$$Q_{SK1} = \frac{\sqrt{R_1 R_2 C_1 C_2}}{R_1 C_1 + R_2 C_1} \tag{4.8}$$

$$Q_{SK2} = \frac{\sqrt{R_3 R_4 C_3 C_4}}{R_3 C_3 + R_4 C_3} \tag{4.9}$$

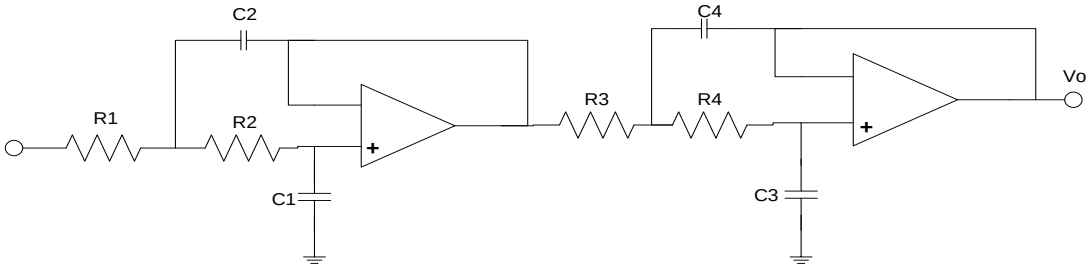
Ardışık yapıların kesim frekansları (ω_{SK1}, ω_{SK2}) 10 krad/sn, kalite faktörleri her bir ardışık yapı için sırasıyla 1/0,7654 ve 1/1,8478 alınmaktadır. Toplam tasarım hatası (J) Denklem (7.10)'da, kesim frekansı sapması ($\Delta\omega$) ve kalite faktörü sapması (ΔQ) Denklem (4.11) ve Denklem (4.12)'de verilmektedir (Bishnu ve ark. 2015).

$$J = 0{,}5(\Delta\omega + \Delta Q) \tag{4.10}$$

$$\Delta\omega = \frac{|\omega_{SK1} - \omega_c| + |\omega_{SK2} - \omega_c|}{\omega_c} \tag{4.11}$$

$$\Delta Q = |Q_{SK1} - \frac{1}{0{,}7654}| + |Q_{SK2} - \frac{1}{1{,}8478}| \tag{4.12}$$

Burada amaç, J hata fonksiyonunu minimize ederek, filtre devresinin optimal devre elemanı değerlerini bulmaktır. Algoritmalar tarafından bulunan hata değeri azaldıkça, filtremiz ideal duruma yakınsayacaktır.



Şekil 4.4. Sallen-Key dördüncü dereceden Butterworth alçak geçiren filtre devresi

4.1.2.2. Örnek tasarım ve analizi

Bu örnek uygulamada, Şekil 4.4'te verilen devre şemasındaki devre elemanı dizilimine uygun olarak dördüncü dereceden Butterworth alçak geçiren filtreyi Sallen-Key topoloji kullanarak, DE, GA, PSO, ACO, HS, VS ve BS algoritmalarıyla tasarımı gerçeklenecektir. Sallen-Key topolojisi kullanılarak tasarlanan Butterworth alçak geçiren analog filtrenin algoritmalar tarafından bulunan devre elemanı değerleri Çizelge 4.3'te verilmektedir. 10 bağımsız çalıştırma sonucu bulunan hatalar Çizelge 4.4'te, bağımsız çalıştırmalar sonucu bulunan en iyi hatanın algoritmalara ait yakınsama grafiği Şekil 4.5'te verilmektedir. Algoritmalar tarafından bulunan filtre devresi devre elemanlarına bağlı olarak PSpice programıyla tasarlanan filtrenin simülasyon grafiği karşılaştırmalı olarak Şekil 4.6'da gösterilmektedir.

Çizelge 4.3. Sallen-Key dördüncü dereceden Butterworth alçak geçiren filtre için algoritmaların bulduğu devre elemanları

Devre Elemanları (E12)	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
R_1(kΩ)	180	3,9	15	3,9	12	15	15
R_2(kΩ)	5,6	12	12	12	15	12	12
C_1(nF)	1	12	6,8	12	6,8	6,8	6,8
C_2(nF)	10	18	8,2	18	8,2	8,2	8,2
R_3(kΩ)	8,2	5,6	8,2	1,8	5,6	5,6	5,6
R_4(kΩ)	5,6	8,2	5,6	12	8,2	8,2	8,2
C_3(nF)	5,6	5,6	5,6	5,6	5,6	5,6	5,6
C_4(nF)	39	39	39	82	39	39	39

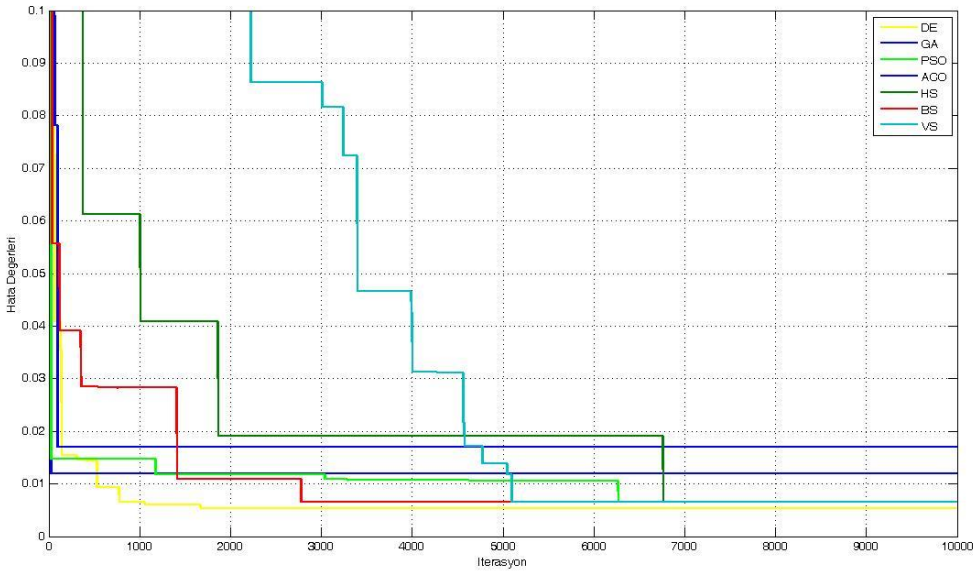
Çizelge 4.4'ten görülebileceği üzere, SV ikinci dereceden alçak filtremizdeki algoritmaların bulduğu (Çizelge 4.2) hatalardan aynı devre elemanı sayısına sahip olmasına rağmen, bu filtre türünde algoritmalarımız daha kötü hatalar bulmaktadır. Bunun sebebi tasarımda kullanılan devre elemanı serisinin (E12), E24 serisine göre daha az devre elemanı değer varyasyonu içermesidir. SV tasarım örneğine benzer olarak, DE algoritması

diğer algoritmalara göre en az tasarım hatasıyla sonuca ulaştığından dolayı, DE algoritmasının her iki tasarım örneğimiz için verimli olarak kullanılabileceği söylenebilmektedir. Minimum en iyi hata değerleri analiz edildiğinde, PSO, HS, VS ve BS algoritmaları aynı en iyi ikinci minimum hata değerlerine ulaşmaktadır. PSO algoritması, belirtilen algoritmalarla aynı hata değerine sahip olmasına rağmen, ortalama hata değerinde bu algoritmalardan daha iyi olması nedeniyle, bu tasarım problemimiz için DE algoritmasından sonra tercih edilebilir bir algoritma olduğu söylenebilmektedir. ACO algoritması tüm hata türlerinde en kötü hataya sahip olduğundan dolayı bu filtre tasarımımızda diğer algoritmalara nazaran iyi bir tasarım sağlayamamaktadır.

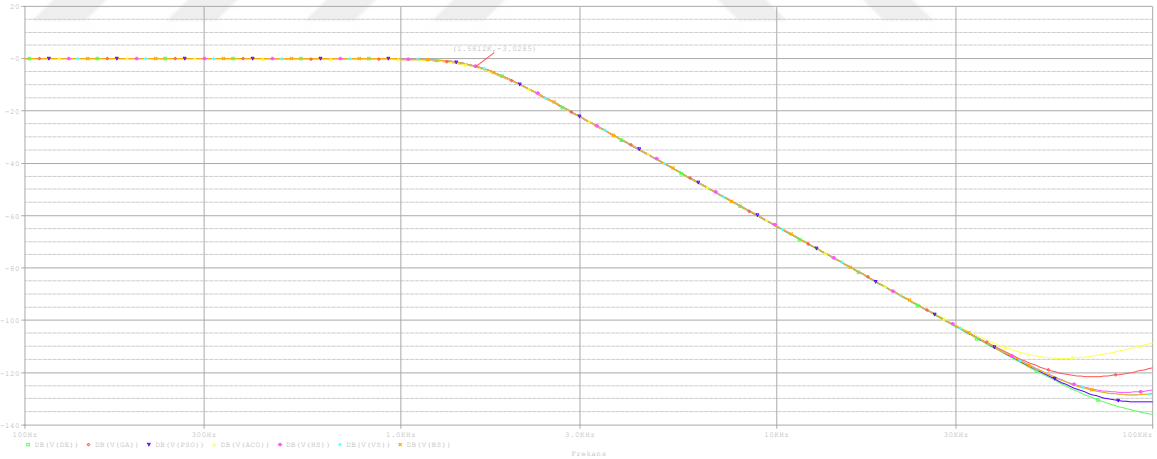
Çizelge 4.4. Sallen-Key dördüncü dereceden Butterworth alçak geçiren filtre için algoritmalar tarafından 10 bağımsız çalıştırmada bulunan hata değerleri

Hata değerleri	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
Minimum en iyi hata	0,0052	0,0120	0,0065	0,0169	0,0065	0,0065	0,0065
Ortalama hata	0,0071	0,0266	0,0085	0,0340	0,0156	0,0152	0,0099
Maksimum en iyi hata	0,0102	0,0445	0,0118	0,0465	0,0305	0,0291	0,0134

Şekil 4.5'ten görüleceği üzere, GA algoritması kendi en iyi çözümüne, SV topolojisiyle tasarlanmış örnekte olduğu gibi en hızlı şekilde yakınsamaktadır. ACO algoritması, kendi en iyi çözümüne en hızlı ikinci yakınsayan algoritma olmasına rağmen (yaklaşık 200 iterasyon), bulunan hata değerleri bakımından en kötü hataya sahip olmaktadır. HS algoritması, SV topoloji tasarım örneğimizde olduğu gibi, SK topolojide de algoritmalar arasından en yavaş hızda çözüme yakınsayan algoritma olarak göze çarpmaktadır. Şekil 4.6'da, bulunan devre elemanı değerlerine bağlı olarak LM741 op-amp makro modeli kullanılarak PSpice programıyla tasarlanan filtremizin AC analiz sonuçları görülmektedir. Filtre devresinin geçirme bandındaki 0 dB kazanç değerinin -3 dB noktasına düşen değer Şekil 4.6'da işaretlenmiş olup kesim frekansı olarak alınmıştır. Algoritmalar, kesim frekansı değerini 9,93 krad/sn olarak bulup, geçirme bandında dalgalanma göstermeyerek, tasarımımızı ideal kesim frekansına yüksek bir yakınsama değeriyle gerçeklemişlerdir.



Şekil 4.5. Sallen-Key dördüncü dereceden Butterworth alçak geçiren filtre için algoritmaların en iyi hata değerine yakınsama grafiği



Şekil 4.6. Sallen-Key dördüncü dereceden Butterworth alçak geçiren filtre için algoritmaların PSpice benzetim grafiği

4.2. Sayısal Filtre Tasarımı

Bu bölümde, günümüzde sıklıkla kullanılan DE, GA, PSO, ACO, HS, VS ve BS algoritmaları ile iki farklı IIR filtre türü ve iki farklı FIR filtre türü ele alınarak, toplamda dört farklı filtre türü tasarlanacaktır. Her bir algoritmanın bulduğu hatalar, yakınsama

grafikleri ve filtre katsayılarına bağlı olarak tasarlanan filtre türü üzerindeki üstünlükleri ve zayıflıkları analiz edilecektir. Her bir filtre türü için, algoritmalar tarafından tasarlanan filtrenin frekans cevabına bağlı olarak, ideal filtreye yakınsaklığı gözlemlenecek ve yapılan tasarımların uygunluğu doğrulanacaktır.

4.2.1. FIR filtre tasarımı

4.2.1.1. Hedef fonksiyon

FIR filtrelerin transfer fonksiyonu, Denklem (4.13)'teki gibi yazılabilir (Shenoi 2005).

$$H(z) = \sum_{k=0}^{N} b_k z^{-k} \tag{4.13}$$

b_k filtre katsayılarını, N filtrenin derecesini, $N+1$ ise FIR filtre katsayıları sayısını temsil eder. FIR filtrenin katsayıları bir vektör şeklinde Denklem (4.14)'teki gibi ifade edilebilir.

$$h[n] = [b_0, b_1 \ldots . b_N] \tag{4.14}$$

Doğrusal faz, tek uzunluklu, FIR filtrenin simetriklik özelliğinden yararlanarak $h(n)$ vektörü;

$$h[n] = h[N-n] \tag{4.15}$$

şeklinde yazılabilir. $h[n]$ vektörünün ayrık Fourier dönüşümü alınarak FIR filtrenin frekans cevabı Denklem (4.16)'daki gibi ifade edilmektedir.

$$H(e^{j\omega_k}) = \sum_{l=0}^{N} h[n] e^{-j\omega_k l} \tag{4.16}$$

Eğer M örnekleme sayısını ifade edecek olursa, $\omega_k = \frac{2\pi k}{M} \in [0, \pi]$ olarak tanımlanır. FIR filtre tasarımlarında kullanılacak hata fonksiyonu IIR filtre tasarımına benzer şekilde, Denklem (4.17)'deki gibi yazılabilir.

$$J = \frac{1}{M} \sum_{\omega} [|H_i(\omega)| - |H_b(\omega)|]^2 \tag{4.17}$$

H_i tasarlanmak istenen filtrenin ideal frekans cevabı, H_b algoritmaların tasarlayacağı filtrenin frekans cevabını ifade eder. Kullanılan algoritmalar, FIR filtre tasarımında *J* hata fonksiyonunu minimize ederek, en uygun FIR filtre katsayılarını, filtrenin simetriklik özelliğinden yararlanarak bulmaktadır.

4.2.1.2. Örnek tasarımlar ve analizleri

FIR filtre tasarımlarında kullanılan örneklere ilişkin başlangıç şartları Çizelge 4.5'te verilmektedir.

Çizelge 4.5. Örnek tasarımlara ilişkin başlangıç şartları

	Örnek Tasarım-1	**Örnek Tasarım-2**
Örnekleme frekansı	1 kHz	10 kHz
Geçirme bandı kesim frekansı	400 Hz	2 kHz
Durdurma bandı kesim frekansı	100 Hz	3.5 kHz
Geçirme bandı dalgalanması	0,1 dB	0,1 dB
Durdurma bandı zayıflaması	30 dB	30 dB
Filtre tipi	Yüksek geçiren filtre	Alçak geçiren filtre
Popülasyon sayısı	50	50
Maksimum iterasyon sayısı	100	100
Bağımsız çalıştırılma sayısı	10	10

İlk örnek uygulamada, Çizelge 4.5'te verilen başlangıç şartlarına uygun olarak (Çizelge 4.5'te ikinci sütun) onuncu dereceden yüksek geçiren filtrenin DE, GA, PSO, ACO, HS, VS ve BS algoritmaları ile tasarımı gerçekleştirilmiştir. FIR filtrenin algoritmalar tarafından bulunan katsayıları Çizelge 4.6'da verilmektedir. 10 bağımsız çalıştırma sonucu bulunan hatalar Çizelge 4.7'da, bağımsız çalıştırmalar sonucu bulunan en iyi hatanın algoritmalara ait yakınsama grafiği Şekil 4.7'de verilmektedir. Algoritmalar tarafından

tasarlanan filtreye ilişkin frekans cevabı grafiği, her bir algoritma için karşılaştırmalı olarak Şekil 4.8'de gösterilmektedir.

Çizelge 4.6. Örnek FIR filtre tasarımı-1 için algoritmaların bulduğu FIR filtre katsayıları

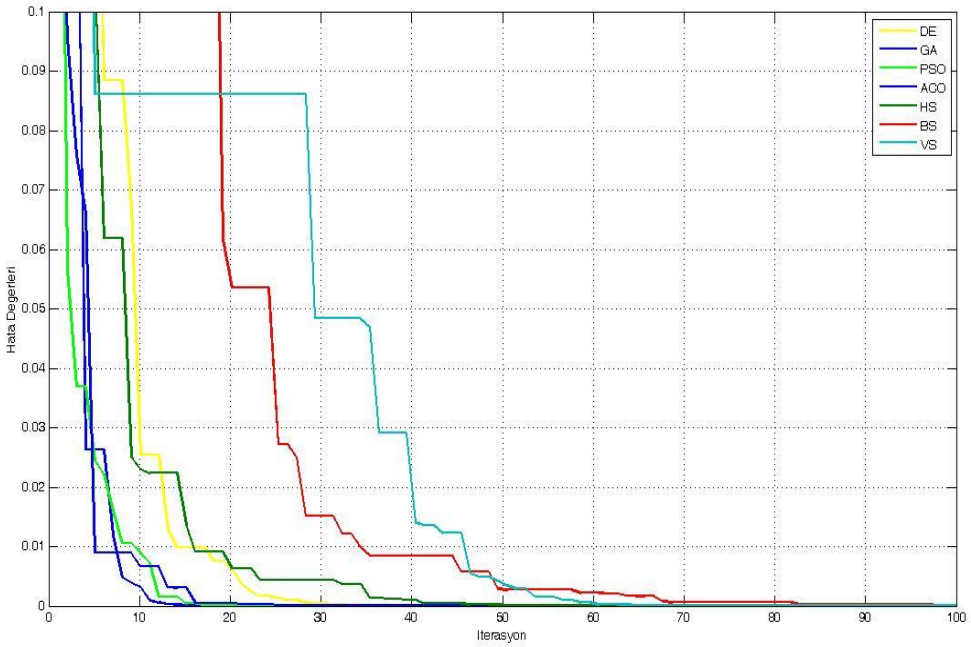
	DE	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
b[1]=b[11]	-0,013966	-0,013964	-0,013965	-0,013963	-0,013926	-0,013963	-0,009975
b[2]=b[10]	-0,026742	-0,026746	-0,026747	-0,026747	-0,026651	-0,026742	-0,029643
b[3]=b[9]	0,056702	0,056708	0,056706	0,056706	0,056584	0,056703	0,056798
b[4]=b[8]	0,054877	0,054873	0,054875	0,054874	0,054880	0,054874	0,053476
b[5]=b[7]	-0,297248	-0,297244	-0,297244	-0,297244	-0,297308	-0,297245	-0,295117
b[6]	0,430736	0,430730	0,430730	0,430730	0,430736	0,430685	0,430350

Çizelge 4.7'deki sonuçlara göre, PSO algoritması hem ortalama hata değerinde hem de en iyi hata değerinde en iyi hataya sahip olmaktadır. PSO algoritmasının bu tasarım örneği için hata değerleri bakımından en uygun algoritma olduğu söylenebilmektedir. GA algoritması ikinci en iyi hataya sahip algoritma olmasına rağmen ortalama tasarım hatasında ACO algoritmasıyla beraber kötü bir performans göstermektedirler. Bu, GA ve ACO algoritmasının farklı çalıştırmalarda, birbirinden çok farklı hatalarda FIR filtreyi tasarlayabileceğini, bu örnek tasarım üzerinde bir kararlılık sağlayamadığını göstermektedir. Diğer yandan, DE algoritması PSO dışındaki tüm algoritmalara göre bu tasarım problemi üzerinde daha az hata dalgalanmasıyla tasarım yapabilmektedir. BS algoritması, tüm hata türleri değerlendirildiğinde bu örnek problem üzerinde en kötü tasarımı yapan algoritma olarak göze çarpmaktadır. Şekil 4.7'de görüldüğü gibi, GA algoritması kendi en iyi hata değerine, yaklaşık 15 iterasyonda en hızlı yakınsamayı sağlamaktadır. Algoritmalar arasında en iyi hata değerine sahip PSO algoritması, hız bakımından algoritmalar arasındaki en iyi çözüme yaklaşık 20 iterasyondan sonra iyi bir yakınsama göstererek, hem hız hem doğruluk bakımından bu örnek tasarım problemi için tercih edilebilir bir algoritma davranışı sergilemektedir. DE algoritması ise 30 iterasyondan sonra kendi en iyi çözümüne yakınsayarak kıyaslanan algoritmalar arasında ortalama bir hıza sahip olduğu söylenebilmektedir. BS algoritması diğer algoritmalar arasında hem

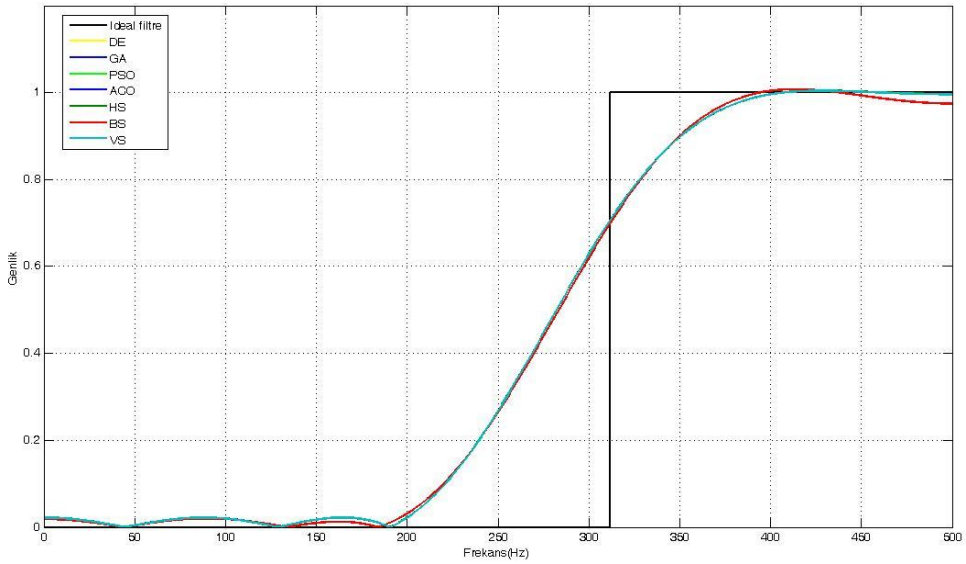
yakınsadığı çözüm bakımından en kötü, hem de hız bakımından en yavaş olduğundan dolayı bu örnek tasarım üzerinde en kötü performansı sergilemektedir. Şekil 4.8'de algoritmaların frekans cevabı grafiğine bakılacak olursa, algoritmalar ideale yakın, düşük hatalarla birbirine çok benzer frekans cevabı vermektedirler.

Çizelge 4.7. Örnek FIR filtre tasarımı-1 için algoritmalar tarafından 10 bağımsız çalıştırmada bulunan hata değerleri

Hata değerleri	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
Minimum en iyi hata	$1{,}516\times10^{-10}$	$9{,}580\times10^{-12}$	$1{,}094\times10^{-14}$	$3{,}731\times10^{-12}$	$5{,}933\times10^{-8}$	$2{,}064\times10^{-9}$	$6{,}054\times10^{-5}$
Ortalama hata	$6{,}834\times10^{-10}$	$2{,}816\times10^{-6}$	$8{,}188\times10^{-14}$	$3{,}570\times10^{-6}$	$2{,}352\times10^{-7}$	$3{,}308\times10^{-7}$	$6{,}474\times10^{-4}$
Maksimum en iyi hata	$1{,}645\times10^{-9}$	$2{,}693\times10^{-5}$	$2{,}866\times10^{-13}$	$3{,}568\times10^{-5}$	$5{,}478\times10^{-7}$	$1{,}409\times10^{-6}$	0,0019



Şekil 4.7. Örnek FIR filtre tasarımı-1 için algoritmaların en iyi hata değerine yakınsama grafiği



Şekil 4.8. Örnek FIR filtre tasarımı-1 için frekans cevabı grafiği

İkinci örnek uygulamada, Çizelge 4.5'te verilen başlangıç şartlarına uygun olarak (Çizelge 4.5'te üçüncü sütun) on ikinci dereceden alçak geçiren filtrenin DE, GA, PSO, ACO, HS, VS ve BS algoritmaları ile tasarımı gerçeklenmiştir. FIR filtrenin algoritmalar tarafından bulunan katsayıları Çizelge 4.8'de verilmektedir. 10 bağımsız çalıştırma sonucu bulunan hatalar Çizelge 4.9'da, bağımsız çalıştırmalar sonucu bulunan en iyi hatanın algoritmalara ait yakınsama grafiği Şekil 4.9'da verilmektedir. Algoritmalar tarafından tasarlanan filtreye ilişkin frekans cevabı grafiği, her bir algoritma için karşılaştırmalı olarak Şekil 4.10'de gösterilmektedir. Çizelge 4.9'dan görüldüğü gibi, birinci örnek tasarıma benzer olarak PSO algoritması diğer algoritmalar arasında hem en iyi hata değerine hem de en iyi ortalama hata değerine, HS algoritması ise en kötü hata değerine ulaşmaktadır. Bu, PSO algoritmasının FIR filtre örnek tasarımları için hata açısından en uygun olduğunu, HS algoritmasının ise diğer algoritmalara nazaran uygun olmadığını göstermektedir. Diğer yandan her iki örnek tasarımı için ortalama hatalara bakıldığında, DE algoritmasının, bağımsız çalıştırmalar boyunca, PSO algoritması dışındaki tüm algoritmalardan daha iyi bir performans ortaya koyduğu söylenebilmektedir. BS algoritması ilk örnek tasarımda olduğu gibi tüm hata türlerinde en başarısız sonuçlara ulaştığından dolayı, uygulanan FIR filtre örnek tasarımlarında tercih edilebilecek algoritmalar arasında yer almamaktadır.

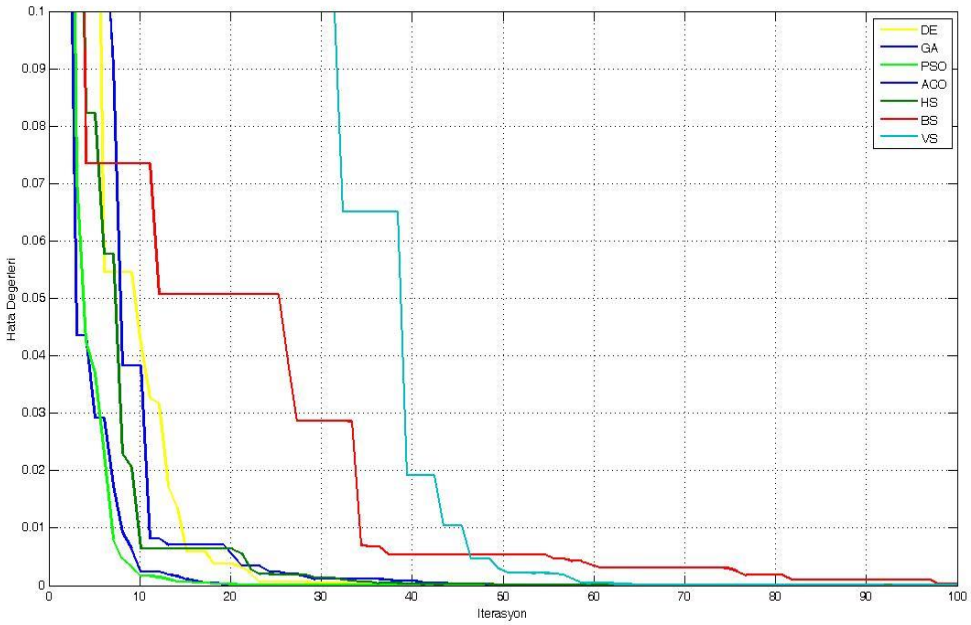
Çizelge 4.8. Örnek FIR filtre tasarımı-2 için algoritmaların bulduğu FIR filtre katsayıları

	DE	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
b[1]=b[13]	-0,013929	-0,0140165	-0,013922	-0,013701	-0,014770	-0,016278	-0,017401
b[2]=b[12]	0,008587	0,008604	0,008590	0,008508	0,008814	0,011961	0,003061
b[3]=b[11]	0,042645	0,042669	0,0426562	0,043091	0,042067	0,038294	0,043658
b[4]=b[10]	-0,068398	-0,068397	-0,068393	-0,068457	-0,068095	-0,063164	-0,073726
b[5]=b[9]	-0,057525	-0,057520	-0,057528	-0,057670	-0,056547	-0,063446	-0,055172
b[6]=b[8]	0,300940	0,300975	0,300936	0,300638	0,300932	0,307280	0,302532
b[7]	0,569853	0,569824	0,569862	0,569650	0,570490	0,563418	0,564435

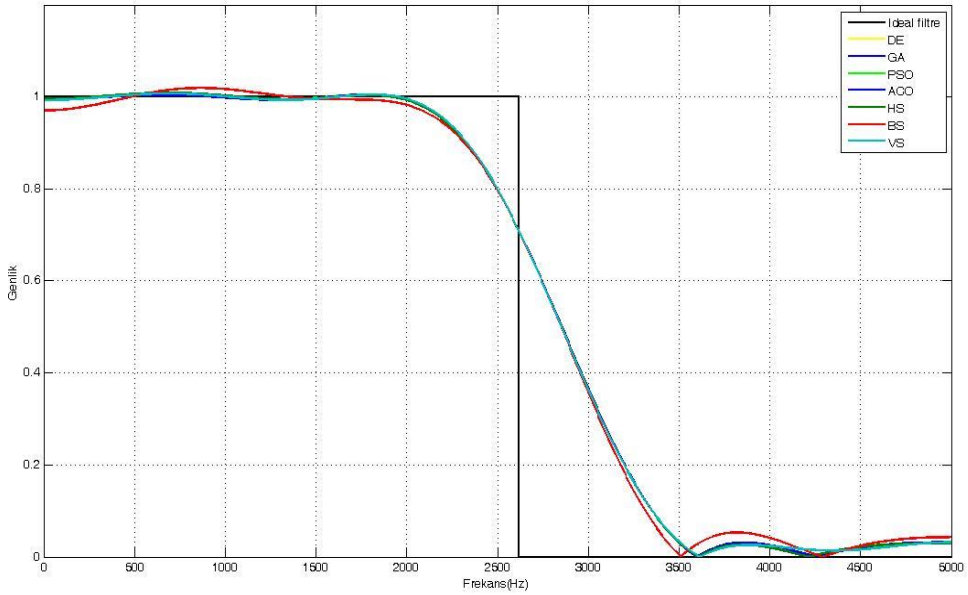
Çizelge 4.9. Örnek FIR filtre tasarımı-2 için algoritmalar tarafından 10 bağımsız çalıştırmada bulunan hata değerleri

Hata değerleri	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
Minimum en iyi hata	$5{,}344\times10^{-10}$	$2{,}301\times10^{-8}$	$1{,}528\times10^{-13}$	$7{,}586\times10^{-7}$	$4{,}640\times10^{-6}$	$1{,}003\times10^{-5}$	$1{,}819\times10^{-4}$
Ortalama hata	$1{,}304\times10^{-6}$	$1{,}506\times10^{-5}$	$1{,}003\times10^{-6}$	$1{,}493\times10^{-5}$	$9{,}453\times10^{-5}$	$4{,}564\times10^{-4}$	$8{,}102\times10^{-4}$
Maksimum en iyi hata	$1{,}003\times10^{-5}$	$6{,}589\times10^{-5}$	$1{,}003\times10^{-5}$	$6{,}614\times10^{-5}$	$2{,}126\times10^{-4}$	0,0018	0,0020

Şekil 4.9'da algoritmaların yakınsama grafiğinden görüleceği üzere, GA ve PSO algoritmaları kendi en iyi çözümlerine birbirine çok yakın iterasyonlarda yakınsayarak, bu tasarım örneğinde kendi en iyi hatalı örneklerini benzer hızda tasarlamaktadırlar. Diğer yandan DE algoritması kendi en iyi hata değerine yakınsamakta diğer algoritmalara göre ortalama bir hız gösterdiği görülmektedir. VS ve BS algoritmaları, ilk FIR filtre örnek tasarım problemine benzer olarak, algoritmalar arasında en yavaş olan ilk iki algoritma olmaktadır. Şekil 4.10'daki frekans cevabı grafiği incelenecek olursa, algoritmalarımız alçak geçiren filtre yapısına uygun, ideal filtreye yakın, düşük hatalı frekans cevabı grafikleri elde etmektedir.



Şekil 4.9. Örnek FIR filtre tasarımı-2 için algoritmaların en iyi hata değerine yakınsama grafiği



Şekil 4.10. Örnek FIR filtre tasarımı-2 için frekans cevabı grafiği

4.2.2. IIR filtre tasarımı

4.2.2.1. Hedef fonksiyon

IIR filtrenin frekans cevabı Denklem (4.18)'de verilmektedir (Shenoi 2005).

$$H(\omega_k) = \frac{\sum_{l=0}^{m} b_1 e^{-j\omega_k l}}{1+\sum_{l=1}^{n} a_1 e^{-j\omega_k l}} \tag{4.18}$$

$\omega_k = \frac{2\pi k}{M}$ ve $H(\omega_k)$, IIR filtrenin transfer fonksiyonudur. [0,π] aralığında frekans, M sayıda örneklenirse;

$$\begin{aligned} H_i(\omega) &= [H_i(\omega_1), H_i(\omega_2) \ldots . H_i(\omega_M)] \\ H_b(\omega) &= [H_b(\omega_1), H_b(\omega_2) \ldots . H_b(\omega_M)] \end{aligned} \tag{4.19}$$

M örnekli frekans cevabı vektörleri elde edilmiş olur. H_i tasarlanmak istenen filtrenin ideal frekans cevabını, H_b algoritmaların tasarlayacağı filtrenin frekans cevabını ifade etmektedir. IIR filtre tasarımlarında kullanılacak hata fonksiyonu Denklem (4.20)'deki gibi yazılabilmektedir.

$$J = \frac{1}{M} \sum_{\omega} [|H_i(\omega)| - |H_b(\omega)|]^2 \tag{4.20}$$

Algoritmaların IIR filtre tasarımındaki amacı, hedef fonksiyon J'yi minimum yapacak şekilde ideale en uygun pay ve payda katsayılarını elde etmektir. Böylelikle ideal duruma en uygun IIR filtre tasarımı gerçekleştirilebilmektedir.

4.2.2.2. Örnek tasarımlar ve analizleri

IIR filtre tasarımlarında kullanılan örneklere ilişkin başlangıç şartları Çizelge 4.9'da verilmektedir.

Çizelge 4.9. Örnek tasarımlara ilişkin başlangıç şartları

	Örnek Tasarım-1	**Örnek Tasarım-2**
Örnekleme frekansı	10 kHz	1 kHz
Geçirme bandı kesim frekansı	3 kHz	200 Hz
Durdurma bandı kesim frekansı	2 kHz	240 Hz
Geçirme bandı dalgalanması	0,1 dB	0,1 dB
Durdurma bandı zayıflaması	30 dB	30 dB
Filtre tipi	Yüksek geçiren filtre	Alçak geçiren filtre
Filtre türü	Butterworth	Chebyshev Tip 1
Popülasyon sayısı	100	100
Maksimum iterasyon sayısı	1000	1000
Bağımsız çalıştırılma sayısı	10	10

İlk örnek uygulamada, Butterworth filtre türünü kullanarak, Çizelge 4.9'da verilen başlangıç şartlarına uygun olarak (Çizelge 4.9'da ikinci sütun) dokuzuncu dereceden yüksek geçiren Butterworth filtrenin, DE, GA, PSO, ACO, HS, VS ve BS algoritmaları ile tasarımı gerçekleştirilmiştir. Algoritmalar tarafından tasarlanan filtrenin pay katsayıları Çizelge 4.10'da, payda katsayıları Çizelge 4.11'de tablolandırılmaktadır. 10 bağımsız çalıştırma sonucu bulunan hatalar Çizelge 4.12'de, bağımsız çalıştırmalar sonucu bulunan en iyi hatanın, algoritmalara ait yakınsama grafiği Şekil 4.11'de verilmektedir. Algoritmalar tarafından tasarlanan filtreye ilişkin frekans cevabı grafiği, her bir algoritma için karşılaştırmalı olarak Şekil 4.12'de gösterilmektedir. Çizelge 4.12'deki sonuçlara göre, DE algoritması örnek tasarımımızda kullandığımız algoritmalar arasında, 10 bağımsız çalıştırmada hem minimum en iyi hataya hem de ortalama hatada en iyi değere ulaşan algoritma olmaktadır. Bu, DE algoritmasının bu örnek problem için diğer algoritmalara göre daha stabil bir tasarım sağladığını göstermektedir.

Çizelge 4.10. Örnek IIR filtre tasarımı-1 için algoritmaların bulduğu pay katsayıları

	DE	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
b[1]	0,007311	0,006801	0,001064	0,002791	0,002254	0,005776	0,003131
b[2]	-0,061158	-0,059225	-0,042018	-0,057425	-0,047045	-0,057216	-0,045554
b[3]	0,226148	0,224362	0,190075	0,234548	0,205157	0,222452	0,197833
b[4]	-0,499922	-0,499979	-0,450946	-0.533125	-0,472005	-0,499547	-0,461320
b[5]	0,730028	0,731789	0,674898	0,782812	0,699087	0,733276	0,688231
b[6]	-0,730028	-0,731789	-0,674898	-0,782812	-0,699087	-0,733276	-0,688231
b[7]	0,499922	0,499979	0,450946	0,533125	0,472005	0,499547	0,461320
b[8]	-0,226148	-0,224362	-0,190075	-0,234548	-0,205157	-0,222452	-0,197833
b[9]	0,061158	0,059225	0,042018	0,057425	0,047045	0,057216	0,045554
b[10]	-0,007311	-0,006801	-0,001064	-0,002791	-0,002254	-0,00577	-0,003131

Çizelge 4.11. Örnek IIR filtre tasarımı-1 için algoritmaların bulduğu payda katsayıları

	DE	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
a[1]	1,0	1,0	1,0	1,0	1,0	1,0	1,0
a[2]	-0,189761	-0,183452	-0,121190	-0,297375	-0,122719	-0,200394	-0,152567
a[3]	1,100744	1,134523	1,055666	1,098311	1,112197	1,104284	1,038771
a[4]	-0,274148	-0,217615	-0,170495	-0,361192	-0,160881	-0,256276	-0,221990
a[5]	0,364453	0,420806	0,314153	0,336703	0,349545	0,375060	0,300821
a[6]	-0,062419	-0,017471	-0,022290	-0,101992	-0,039428	-0,044113	-0,040092
a[7]	0,058758	0,075434	0,037281	0,022354	0,024616	0,058975	0,038361
a[8]	0,005045	0,008328	0,003728	-0,012250	-0,016265	0,004989	0,004805
a[9]	0,003883	0,001951	0,000101	0,000381	0,000101	0,002751	$1{,}1\times10^{-4}$
a[10]	-0,000119	-0,000752	-0.000004	-0,000231	$-1{,}0\times10^{-5}$	$-2{,}8\times10^{-4}$	$-1{,}0\times10^{-5}$

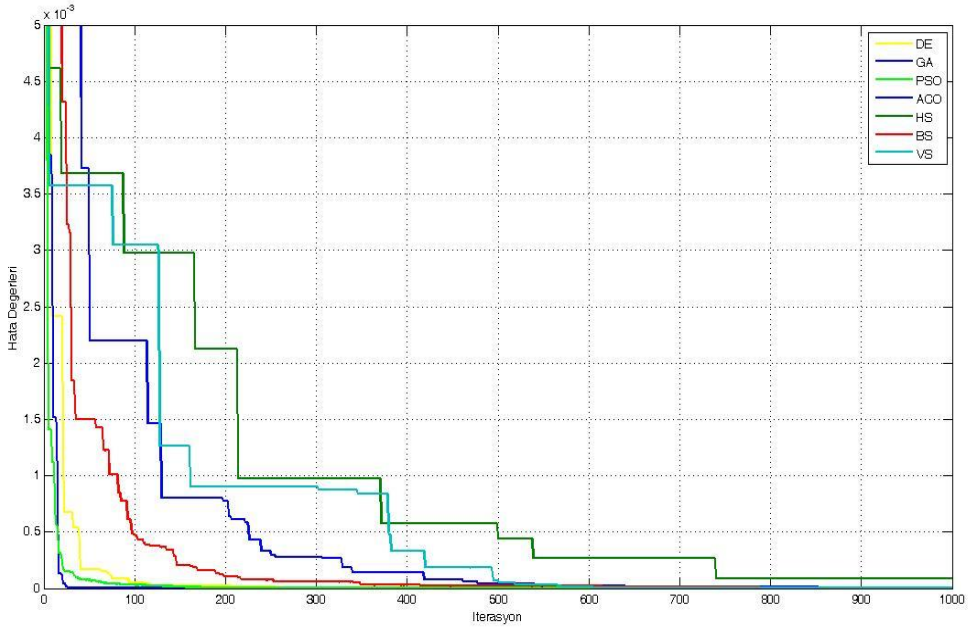
Çizelge 4.12'den, PSO algoritmasını en iyi bulduğu hata değerine göre diğer algoritmalar ile kıyaslandığında, bu algoritmanın en iyi performansa sahip algoritmalardan biri olup, ortalama hata değerlerine göre de bağımsız çalıştırmalarda başarılı tasarımlar yapabileceği

söylenebilmektedir. Diğer yandan, HS algoritması, bulduğu en iyi ve ortalama hata değerleri açısından, kullanılan algoritmalarla kıyaslandığında bu tasarım problemi için uygun bir çözüm taşımamaktadır.

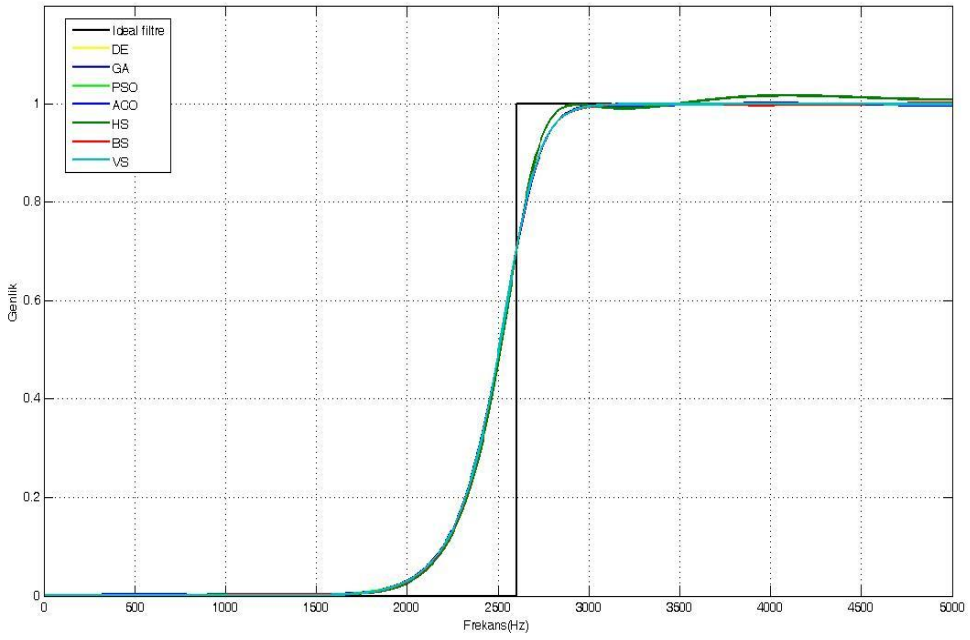
Çizelge 4.12. Örnek IIR filtre tasarımı-1 için algoritmalar tarafından 10 bağımsız çalıştırmada bulunan hata değerleri

Hata değerleri	**DE**	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
Minimum en iyi hata	$1{,}997\times10^{-8}$	$3{,}928\times10^{-7}$	$5{,}483\times10^{-7}$	$4{,}683\times10^{-6}$	$8{,}860\times10^{-5}$	$3{,}431\times10^{-7}$	$1{,}765\times10^{-6}$
Ortalama hata	$8{,}192\times10^{-8}$	$1{,}579\times10^{-5}$	$2{,}468\times10^{-6}$	$4{,}544\times10^{-5}$	$2{,}019\times10^{-4}$	$1{,}309\times10^{-5}$	$7{,}106\times10^{-6}$
Maksimum en iyi hata	$2{,}769\times10^{-7}$	$5{,}525\times10^{-5}$	$7{,}024\times10^{-6}$	$2{,}054\times10^{-4}$	$3{,}146\times10^{-4}$	$3{,}218\times10^{-5}$	$1{,}483\times10^{-5}$

Şekil 4.11'den görüleceği üzere, GA algoritması en iyi hata değerine yaklaşık 50. iterasyondan sonra iyi bir yakınsama sağlayarak algoritmalar arasında en hızlı şekilde ulaşmaktadır. Buna rağmen yakınsadığı minimum en iyi hata değeri algoritmalar arasında en iyi değer olmamaktadır. Bu, GA algoritmasının örnek tasarım üzerinde hata bakımından kötü, hız bakımından en iyi sonuca ulaştığını göstermektedir. Diğer yandan, PSO ve DE algoritmaları minimum en iyi hata değerlerine yaklaşık 150. iterasyondan sonra iyi bir yakınsama sağlamaktadır. Bu, algoritmaların hız-hata dengesi bakımından iyi bir yaklaşım sağladığını ve her iki kriter göz önünde bulundurulduğunda tercih edilebilir olduğunu göstermektedir. HS algoritmasının bulduğu hata değeri ve yakınsama hızı diğer algoritmalardan kötü olduğundan dolayı hem hız hem hata bakımından bu problem için tercih edilmeyebilmektedir. Şekil 4.12'den yola çıkarak, algoritmaların frekans cevaplarının ideal alçak geçiren filtreye uygun olarak iyi bir yaklaşım sergilediği söylenebilmektedir.



Şekil 4.11. Örnek IIR filtre tasarımı-1 için algoritmaların en iyi hata değerine yakınsama grafiği



Şekil 4.12. Örnek IIR filtre tasarımı-1 için frekans cevabı grafiği

İkinci örnek uygulamada, Chebyshev filtre türünü kullanarak, Çizelge 4.9'da verilen başlangıç şartlarına uygun olarak (Çizelge 4.9'da üçüncü sütun) dokuzuncu dereceden alçak geçiren Chebyshev filtrenin, DE, GA, PSO, ACO, HS, VS ve BS algoritmaları ile tasarımı gerçeklenmiştir. Algoritmalar tarafından tasarlanan filtrenin pay katsayıları Çizelge 4.13'te, payda katsayıları Çizelge 4.14'te verilmektedir. 10 bağımsız çalıştırma sonucu bulunan hatalar Çizelge 4.15'te, bağımsız çalıştırmalar sonucu bulunan en iyi hatanın, algoritmalara ait yakınsama grafiği Şekil 4.13'te gösterilmektedir. Algoritmalar tarafından tasarlanan filtreye ilişkin frekans cevabı grafiği, her bir algoritma için karşılaştırmalı olarak Şekil 4.14'te verilmektedir. Çizelge 4.15'teki sonuçlara göre, DE algoritması minimum ve maksimum en iyi hata ve ortalama hata değerlerine göre diğer algoritmalardan daha iyi sonuçlara ulaşmaktadır. Tüm hata türlerinde bu algoritmanın en iyi sonuca ulaşması bize yüksek geçiren filtre tasarımı yaptığımız bu örnekte, diğer kullanılan algoritmalardan daha uygun olduğunu doğrulamaktadır. Diğer yandan, DE algoritması alçak geçiren filtre tasarım örneğinde de iyi sonuç verdiğinden, her iki tasarım uygulaması için kullanılabilir olduğunu göstermektedir.

Çizelge 4.13. Örnek IIR filtre tasarımı-2 için algoritmaların bulduğu pay katsayıları

	DE	**GA**	**PSO**	**ACO**	**HS**	**VS**	**BS**
b[1]	0,000899	0,000899	0,000900	0,000872	0,000900	0,000743	0,000900
b[2]	0,002999	0,003001	0,003000	0,002992	0,002498	0,002791	0,003000
b[3]	0,008999	0,006387	0,007156	0,007471	0,004887	0,006511	0,006214
b[4]	0,014679	0,015252	0,015725	0,013178	0,016680	0,014848	0,014868
b[5]	0,024378	0,021386	0,022539	0,022326	0,020143	0,021638	0,021221
b[6]	0,024378	0,021386	0,022539	0,022326	0,020143	0,021638	0,021221
b[7]	0,014679	0,015252	0,015725	0,013178	0,016680	0,014848	0,014868
b[8]	0,008999	0,006387	0,007156	0,007471	0,004887	0,006511	0,006214
b[9]	0,002999	0,003001	0,003000	0,002992	0,002498	0,002791	0,003000
b[10]	0,000899	0,000899	0,000900	0,000872	0,000900	0,000743	0,000900

Çizelge 4.14. Örnek IIR filtre tasarımı-2 için algoritmaların bulduğu payda katsayıları

	DE	GA	PSO	ACO	HS	VS	BS
a[1]	1,0	1,0	1,0	1,0	1,0	1,0	1,0
a[2]	-4,04131	-4,12383	-4,07660	-4,12204	-4,12819	-4,10672	-4,13696
a[3]	9,21007	9,51332	9,33488	9,49158	9,51049	9,42817	9,56240
a[4]	-14,10541	-14,73795	-14,36178	-14,66492	-14,71410	-14,52612	-14,84481
a[5]	15,72436	16,60060	16,07466	16,4622	16,54926	16,25758	16,75849
a[6]	-13,04141	-13,90881	-13,38472	-13,73566	-13,84942	-13,51846	-14,07833
a[7]	8,01188	8,63168	8,25535	8,48175	8,58981	8,31290	8,76534
a[8]	-3,51332	-3,82487	-3,63506	-3,73586	-3,80803	-3,64185	-3,90052
a[9]	1,00315	1,10430	1,04254	1,07073	1,10148	1,03599	1,13251
a[10]	-0,14414	-0,16066	-0,15053	-0,15425	-0,16035	-0,14756	-0,16622

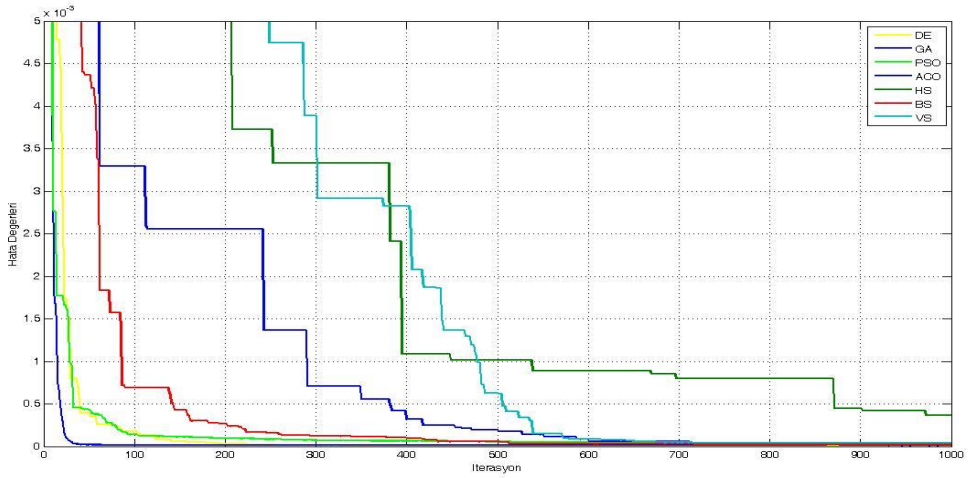
Çizelge 4.15'ten, GA, PSO ve BS algoritmaları birbirine yakın fakat DE algoritmasına göre daha kötü hata sonuçları elde ettiğinden dolayı DE algoritmasına nazaran iyi bir seçenek olmamaktadırlar. HS algoritması minimum en iyi ve ortalama hata değerleri göz önünde bulundurulduğunda kötü bir performans sergilediğinden dolayı bu tasarım örneğimiz için tercih edilen algoritmalar arasında olmamaktadır.

Çizelge 4.15. Örnek IIR filtre tasarımı-2 için algoritmalar tarafından 10 bağımsız çalıştırmada bulunan hata değerleri

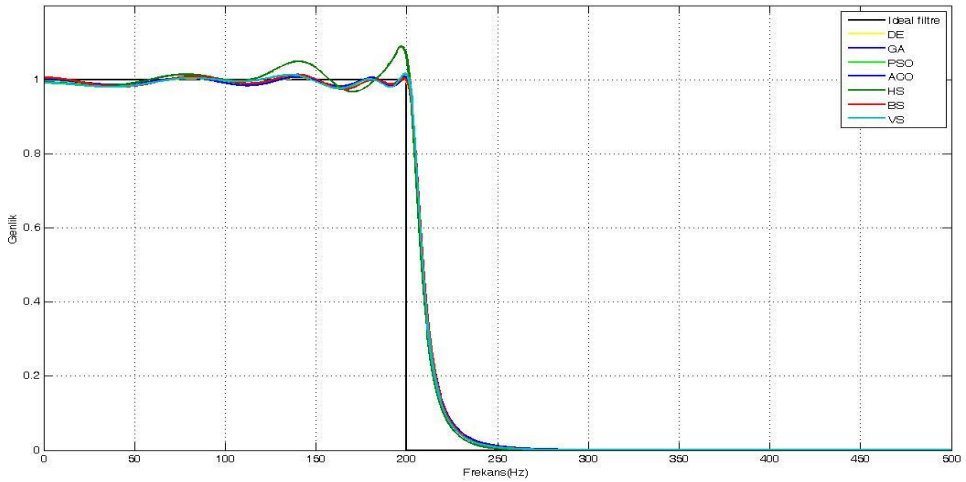
	DE	GA	PSO	ACO	HS	VS	BS
Minimum en iyi hata	$5{,}589\times10^{-6}$	$1{,}593\times10^{-5}$	$1{,}684\times10^{-5}$	$1{,}220\times10^{-5}$	$3{,}691\times10^{-4}$	$4{,}205\times10^{-5}$	$2{,}285\times10^{-5}$
Ortalama hata	$5{,}769\times10^{-6}$	$6{,}170\times10^{-5}$	$4{,}753\times10^{-5}$	$1{,}772\times10^{-4}$	$5{,}698\times10^{-4}$	$2{,}337\times10^{-4}$	$3{,}246\times10^{-5}$
Maksimum en iyi hata	$6{,}912\times10^{-6}$	$1{,}573\times10^{-4}$	$1{,}614\times10^{-4}$	$5{,}171\times10^{-4}$	$8{,}080\times10^{-4}$	$4{,}843\times10^{-4}$	$5{,}321\times10^{-5}$

Şekil 4.13'te görüldüğü üzere, PSO ve GA algoritmalarının ulaştıkları en iyi çözüm birbirine benzer olmasına rağmen, GA algoritması kendi en iyi çözümüne en hızlı şekilde yakınsamaktadır. DE algoritması ise bulduğu en iyi çözüm GA algoritmasından daha iyi

olmasına rağmen, çözüme yakınsama hızı daha yavaş olmaktadır. VS algoritması en iyi hata değerine yaklaşık 500 iterasyonda yakınsayarak en yavaş ikinci algoritma olmakta ve yakınsadığı hata değeri algoritmalar arasında en iyi minimum hata değeri olmamaktadır. Diğer yandan, HS algoritması hem bulduğu en iyi çözüm açısından hem de yakınsama hızı açısından algoritmalar arasında en kötü performansı göstermektedir. Bu, her iki tasarım örneği için HS algoritmasının iyi bir seçim olmadığını göstermektedir. Şekil 4.14'ten, algoritmaların ideal alçak geçiren filtre türüne başarılı bir yaklaşım sergilediği görülmektedir.



Şekil 4.13. Örnek IIR filtre tasarımı-2 için algoritmaların en iyi hata değerine yakınsama grafiği



Şekil 4.14. Örnek IIR filtre tasarımı-2 için frekans cevabı grafiği

5. SONUÇLAR VE TARTIŞMA

Filtreler elektrik, elektronik, bilgisayar vb. bilimlerde sıklıkla kullanılan bileşenlerdendir. Genel olarak filtreleme işlemi bazı işaretlerin geçmesini, bazı işaretlerin durdurulmasını, aynı ortamda bulunan birden fazla işaretin ayrıştırılması ve gürültü gibi istenmeyen etkilerin azaltmasını veya yok edilmesini sağlamak amacıyla kullanılan metotlar bütünüdür. Filtreler genel olarak kendi içerisinde sayısal ve analog olarak iki ana dala ayrılmaktadır. Sayısal filtreler FIR ve IIR filtreler olarak kendi içlerinde ayrılırken, analog filtreler tasarlandığı topolojinin ismiyle adlandırılabilmektedir.

Tez çalışmasının birinci kısmında, analog filtre tasarımları literatürde sıklıkla kullanılan GA, DE, PSO, ACO, HS, VS ve BS algoritmaları vasıtasıyla gerçeklenmektedir. Algoritmalar alçak geçiren aktif analog filtreyi, durum değişkenli ve Sallen-Key filtre topolojilerini kullanarak tasarlamışlardır. Tasarımların gerçeklenmeye uygunluğu açısından algoritmalar tarafından bulunan devre elemanları endüstri üretim serilerinden seçilmektedir. Bu, devre elemanlarının bulunan değerlerini paralel/seri bağlama gibi yöntemlerle elde etmek yerine ilişkili devre elemanını doğrudan kullanmayı olanaklı kılmaktadır. Algoritmalar tarafından bulunan devre elemanlarına göre, PSpice programı aracılığıyla devre tasarlanmış ve ilgili benzetim grafiklerinden tasarlanan devrelerin doğrulanmıştır. Gerçekleştirilen analog filtre tasarım örneklerinde, her iki topoloji için DE algoritmasının $1{,}857\times10^{-5}$ ve 0,0052 minimum en iyi hata değerleriyle algoritmalar arasında en iyi performansı göstermektedir.

Bu tez çalışmasının ikinci kısmında sayısal filtre tasarımları belirtilen evrimsel algoritmalar ile gerçekleştirilmiştir. Sayısal filtre türlerinden FIR ve IIR filtrelerin her biri için iki farklı örnek çalışma ele alınmıştır. Bu örnek çalışmalarda alçak ve yüksek geçiren filtrelerin optimal filtre katsayıları, belirtilen evrimsel algoritmalar aracılığıyla bulunmuş, algoritmalara ait hata performansları analiz edilmiş, tasarımın doğruluğu ilişkili algoritmanın frekans cevap grafiği ile gözlemlenmiştir. FIR filtre tasarım örneklerinde, PSO algoritması örnek tasarım-1 için $1{,}094\times10^{-14}$ ve örnek tasarım-2 için $1{,}528\times10^{-13}$ minimum en iyi hata değerleriyle algoritmalar arasında en iyi performansı göstermektedir.

IIR filtre tasarım örneklerinde ise DE algoritması algoritmalar arasındaki en iyi performansa, örnek tasarım-1 için $1{,}997\times10^{-8}$ ve örnek tasarım-2 için $5{,}589\times10^{-6}$ hata değerleriyle ulaşmaktadır.

Gerçekleştirilen tez çalışmasında, sayısal ve analog filtrelerin evrimsel algoritmalarla hızlı, etkin ve verimli olarak tasarlanabildiği gözlemlenmektedir. Özellikle analog filtre tasarımlarında karmaşık hesaplama ve tasarım adımları gerektirmeden, istenilen endüstriyel serilerdeki devre elemanları kolaylıkla ve yüksek doğrulukla belirlenebilmektedir. Gelecek çalışmalarda, farklı evrimsel algoritmaların bu alanda kullanılması ve etkinliklerinin araştırılması hedeflenmektedir. Ayrıca kullanılan evrimsel algoritmaların sadece filtre devrelerine değil diğer alanlardaki devre ve sistemlerde performans analizlerinin yapılması planlanmaktadır.

KAYNAKLAR

Allaoua, B., Laoufı, A., Gasbaoui, B., Abderrahmani A. 2009. Setting up PID dc motor speed control alteration parameters using particle swarm optimization strategy. *Leonardo Electronic Journal of Practices and Technologies*, 14: 19-32.

Anonim, 2008. Signal Processing: Continuous and Discrete. http://ocw.mit.edu/courses/mechanical-engineering/2-161-signal-processing-continuous-and-discrete-fall-2008/study-materials /lpopamp.pdf-(Erişim tarihi: 03.06.2016).

Anonim, 2012. Osilatörler ve filtre devreleri. http://megep.meb.gov.tr/mte_program_modul/moduller_pdf/Osilat%C3%B6rler%20Ve%20Filtre%20Devreleri.pdf--(Erişim tarihi: 13.05.2016).

Arumugam, M.S., Rao, M.V.C. 2006. On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. *Discrete Dyn. Nat. Soc.*, DOI: 10.1155/DDNS/2006/79295.

Baher, Hussein. 2012. Signal processing and integrated circuits, Wiley, USA, 472 pp.

Bakshi, U.A., Godse, A.P., Bakshi, A.V. 2011. Linear integrated circuits and applications. Technical Publications, USA, 652 pp.

Batık, Z. 2011. Sayısal filtre tasarım yöntemleri ve performans analizleri. *Yüksek Lisans Tezi*, SAÜ Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Eğitimi Anabilim Dalı, Sakarya.

Bishnu, P.D., Kar, R., Mandal, D., Ghoshal, S.P. 2015. Optimal selection of components value for analog active filter design using simplex particle swarm optimization. *Int. J. Mach. Learn. Cyb.*, 6: 621–636.

Boudjelaba, K., Ros, F., Chikouche, D. 2014. Potential of particle swarm optimization and genetic algorithms for FIR filter design. *Circuits Syst. Signal Process*, 33(10): 3195-3222.

Chen, S., Luk, B.L. 2010. Digital IIR filter design using particle swarm optimization. *International Journal of Modelling, Identification and Control*, 9(4): 327–335.

Civicioglu, P. 2013. Backtracking search optimization algorithm for numerical optimization problems. *Appl. Math. Comput*., 219:8121-8144.

Colin, R.R., Jonathan, E. R. 2003. Genetic algorithms-principles and perspectives, a guide to GA theory. Kluwer Academic Publishers, USA, 345 pp.

Çetinkaya, M.B. 2010. Yapay arı koloni algoritmasıyla sayısal süzgeç tasarımı. *Doktora Tezi*, Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Elektronik Mühendisliği Anabilim Dalı, Kayseri.

Digbalay, B., Subhodip, B., Athanasios, V.V., Sougata, L. 2014. Optimal filter design using an improved artificial bee colony algorithm. *Inf. Sci.*, 281: 443-461.

Doğan, B. Ölmez, T. 2015. A new metaheuristic for numerical function optimization: vortex search algorithm. *Inf. Sci.*, 293: 125-145.

Dorigo, M. 1992. Optimization, Learning and Natural Algorithms. *PhD thesis*, Politecnico di Milano, Italy.

Dorigo, M., Gambardella, L.M. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE T. Evolut. Comput.*, 1: 53-66.

Eberhart, R.C., Shi, Y. 2000. Comparing inertia weights and constriction factors in particle swarm optimization. IEEE Congress Evolutionary Computation, San Diego.

Eberhart, R.C., Shi, Y. 2001. Tracking and optimizing dynamic systems with particle swarms. IEEE Proceedings of Evolutionary Computation, Seoul.

Ertürk, S. 2009. Sayısal işaret işleme. Birsen Yayınevi, İstanbul, Türkiye, 293 s.

Feng, Y., Teng, G.F., Wang, A.X., Yao, Y.M. 2007. Chaotic inertia weight in particle swarm optimization. IEEE Innovative Computing. Information and Control, Kumamoto, Japan.

Geem, Z.W., Kim, J.H., Loganathan, G.V. 2001. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2): 60–68.

Holland, J. 1975. Adaptation in natural and artificial systems. MIT Press, USA 211 pp.

Haykin, S. 2013. Adaptive filter theory. Pearson, USA, 912 pp.

Hercules, G.D. 2012. Analog electronic filters theory, design and synthesis. Springer, USA, 592 pp.

Horrocks, D.H., Spittle, M.C. 1993. Component value selection for active filters using genetic algorithms. Proceedings of IEE/IEEE Workshop on Natural Algorithms in Signal Processing, Chelmsford, UK.

Hrstka, O., Kucerova, A. 2004. Improvement of real coded genetic algorithms based on differential operators preventing premature convergence, *Adv. Eng. Softw.*, 35: 237-246.

Hu, X., 2010. PSO Tutorial. http://www.swarmintelligence.org/tutorials.php-(Erişim tarihi: 06.06.2016).

Kacelenga, R.V., Graumann, P.J., Turner, L.E. 1990. Design of digital filters using simulated annealing. IEEE International Symposium on Circuits Systems, 1-3 May, 1990, New Orleans, USA.

Kalinli, A. 2004. Optimal circuit design using immune algorithm. *Lecture Notes in Computer Science*, 42–52.

Kalinli, A. 2006. Component value selection for active filters using parallel tabu search algorithm. *AEU – Int. J. Electron. Commun.*, 60: 85–92.

Karaboğa, D. 2011. Yapay zeka optimizasyon algoritmaları, Ankara, Türkiye, 232 s.

Karaboga, N. 2005. Digital IIR filter design using differential evolution algorithm. *Eurasip Journal on Applied Signal Processing*, 8: 1269-1276.

Karaboga, N., Cetinkaya, B. 2006. Design of digital FIR filters using differential evolution algorithm. *Circuits Syst. Signal Process*., 25(5): 649-660.

Kaur, K., Dhillon, J.S. 2014. Design of digital IIR filters using integrated cat swarm optimization and differential evolution. *International Journal of Computer Applications*, 99(4): 28–43.

Kayran, A.H., Ekşioğlu, E.M. 2004. Bilgisayar uygulamalarıyla sayısal işaret işleme, Birsen Yayınevi, İstanbul, Türkiye, 347 s.

Kennedy, J., Eberhart, R. 1995. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Perth, Australia.

Lacanette, K., 2010. A Basic introduction to filters-active, passive, and switched-capacitor. Texas Instruments, USA.

Lee K.S., Geem Z.W. (2004). A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice. *Comput. Method Appl. Mech. Eng*., 194: 3902–3933.

Lui, G., Li, Y.X., He, G. 2010. Design of digital FIR filters using differential evolution algorithm based on reserved gene. IEEE Congress on Evolutionary Computation, 18-23 July, 2010.

Mancini, R. 2002. Op Amps for Everyone, Texas Instruments Incorporated, USA, 464 pp.

Mayer, D.G., Kinghorn, B.P, Archer, A.A. 2005. Differential evolution - an easy and efficient evolutionary algorithm for model optimization. *Agr. Syst.*, 83(3): 315-328.

Min, J., Zhenkun, Y., Zhaohui, G. 2007. Optimal components selection for analog active filters using clonal selection algorithms. *Lecture Notes in Computer Science,* 950–959.

Mitra, S.K. 2001. Digital signal processing: a computer-based approach. McGraw-Hill Companies, USA, 896 pp.

Reeves, C.R. 1993. Modern heuristic techniques for combinatorial problems. Wiley, USA, 320 pp.

Saha, S., Kar, R., Mandal, D., Ghoshal, S. 2015. Optimal IIR filter design using gravitational search algorithm with wavelet mutation. *Journal of King Saud University Computer and Information Sciences*, 27(1): 25–39.

Schlichthatle, D. 2000. Digital filters basics and design. Springer, Germany, 361 pp.

Shenoi, B.A. 2005. Introduction to digital signal processing and filter design Wiley, USA, pp. 440.

Shi, Y., Eberhart, R. 1998. A modified particle swarm optimizer. IEEE World Congress on Computational Intelligence, USA.

Simon, D. 2013. Evolutionary optimization algorithms. Wiley, New Jersey, USA, 776 pp.

Singh, B., Dhillon, J.S., Brar, Y.S. 2013. A hybrid differential evolution method for the design of IIR digital filter. *ACEEE International Journal of Signal & Image Processing*, 4: 1-10.

Storn, R, Price, K. 1995. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. International Computer Science Institute.

Storn, R., Price, K. 1997. Differential evolution-a simple and efficient heuristic strategy for global optimization over continuous spaces, *J. Global Optim.*, 11: 341-359.

Tamer, S., Karakuzu, C. 2006. Parçacık sürü optimizasyon algoritması ve benzetim örnekleri. Elektrik-Elektronik ve Bilgisayar Sempozyumu-ELECO, Bursa.

Tsai, C.W., Huang, C.H., Lin, C.L. 2009. Structure-specified IIR filter and control design using real structured genetic algorithm. *Appl. Soft Comput.*, 9(4): 1285–1295.

Vaseghi, S.V. 2009. Advanced digital signal processing and noise reduction. Wiley, USA, 544 pp.

Vural, R.A, Bozkurt, U., Yildirim, T. 2013. Analog active filter component selection with nature inspired metaheuristics. *AEU – Int. J. Electron. Commun.*, 67: 197–205.

Vural, R.A., Yildirim, T. 2010. Component value selection for analog active filter using particle swarm optimization. The 2nd International Conference on Computer and Automation Engineering, Singapore.

Vural, R.A., Yildirim, T., Kadioglu, T., Basargan A. 2012. Performance evaluation of evolutionary algorithms for optimal filter design. *IEEE T. Evolut. Comput.*, 16: 135-147.
Winder, S. 2002. Analog and digital filter design. Newnes, USA, 450 pp.

Yang, X.S. 2010. Engineering optimization: an introduction with metaheuristic applications. Wiley, New Jersey, USA, 377 pp.

Yu, Y., Xinjie, Y. 2007. Cooperative coevolutionary genetic algorithm for digital IIR filter design. *IEEE T. Ind. Electron.,* 54(3): 1311–1318.

ÖZGEÇMİŞ

Adı Soyadı : Yiğit Çağatay Kuyu
Doğum Yeri ve Tarihi : Samsun 08/02/1989
Yabancı Dili : İngilizce

Eğitim Durumu (Kurum ve Yıl)
Lise : Samsun Anadolu Lisesi (2003-2007)
Lisans : Uludağ Universitesi (2008-2013)

Çalıştığı Kurum/Kurumlar ve Yıl : 2015- U.Ü. E. E. Müh. (Araştırma Görevlisi)
İletişim (e-posta) : yigitkuyu@uludag.edu.tr

Yayınları :

Kuyu, Y.Ç., Vatansever F. 2013. Design of virtual mouse based on image processing. 7th International Advanced Technologies Symposium, 30 October-1 November 2013, Istanbul, Turkey.

Vatansever, F., Yalcin, N.A., Kuyu Y.Ç. 2015. Evrimsel algoritmalarla tristörlü doğrultucu devrelerindeki tetikleme açılarının hesaplanması. *Uludağ Üniversitesi Mühendislik Fakültesi Dergisi*, 20(2): 67-77.

Kuyu, Y.Ç., Yalcin N.A., Vatansever, F. 2015. The FIR filter simulator based on evolutionary algorithm. 3rd International Symposium on Innovative Technologies in Engineering and Science, 3-5 June 2015, Valencia, Spain.

Vatansever, F., Yalcin, N.A., Kuyu, Y.Ç, 2015. The design of de-noising simulator. 3rd International Symposium on Innovative Technologies in Engineering and Science, 3-5 June 2015, Valencia, Spain.

Kuyu Y.Ç., Vatansever, F. 2016. The simulator design of FET amplifier circuits. 1st International Conference on Engineering Technology and Applied Sciences, 21-22 April 2016, Afyonkarahisar, Turkey.

Vatansever, F., Kuyu, Y.Ç. 2016. Determination of oscillator circuit components with evolutionary algorithms. 1st International Conference on Engineering Technology and Applied Sciences, 21-22 April 2016, Afyonkarahisar, Turkey.