FAZ DİZİLİ DAİRESEL MİKROŞERİT ANTENLERİN GENETİK ALGORİTMA İLE OPTİMİZASYONU

Mustafa Selim CAN

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2010
ANKARA

Mustafa Selim CAN tarafından hazırlanan "FAZ DİZİLİ DAİRESEL MİKROŞERİT ANTENLERİN GENETİK ALGORİTMA İLE OPTİMİZASYONU" adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım

Doç. Dr. Erkan AFACAN .....
Tez Danışmanı, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Prof. Dr. Erdem YAZGAN .....
Elektrik-Elektronik Mühendisliği, Hacettepe Üniversitesi

Doç. Dr. Erkan AFACAN .....
Elektrik-Elektronik Mühendisliği, Gazi Üniversitesi

Yrd. Doç. Dr. Nursel AKÇAM .....
Elektrik-Elektronik Mühendisliği, Gazi Üniversitesi

Tarih: 07/07/2010

Bu tez ile G.Ü. Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onamıştır.

Prof. Dr. Bilal TOKLU .....
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Mustafa Selim CAN

FAZ DİZİLİ DAİRESEL MİKROŞERİT ANTENLERİN GENETİK ALGORİTMA İLE OPTİMİZASYONU

(Yüksek Lisans Tezi)

Mustafa Selim CAN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Temmuz 2010

ÖZET

Günümüzde sinyal kaynaklarının çoğalmasıyla birlikte atmosferdeki elektromanyetik kirliliğin artması, arzulanan kalitede sinyal elde edilmesini zorlaştırmaktadır. Bu nedenle, haberleşme sistemlerinde anten dizilerinin önemi giderek artmaktadır. Çünkü anten dizileri, tekli antenlerin aksine, istenilen ışıma örüntüsünün gerekli parametreleri değiştirilerek elde edilebilmesini sağlamakta ve endüstriyel uygulamalarda tercih edilmektedir. Anten dizilerinin hedeflenen bir doğrultudan gelen işaretleri maksimum kazançla alma ve istenmeyen doğrultulardan gelen girişim kaynaklarını engelleme özellikleri, anten dizilerini haberleşmede önemli bir çalışma alanı kılmaktadır.

Bu çalışmada, dairesel mikroşerit antenler kullanılarak oluşturulan anten dizileri ele alınmış, bu anten dizilerinin istenilen ışıma örüntüsüne sahip olması için optimizasyon çalışmaları yapılmıştır. Genetik biliminden esinlenerek ortaya atılan ve analitik çözümü olmayan karmaşık problemlerde başarılı sonuçlar veren genetik algoritma yöntemi optimizasyon metodu olarak kullanılarak; genetik algoritma yönteminin, antenin ana hüzmesinin istenilen doğrultuya

yönlendirilmesi için gereken parametre değerlerini elde etmek için uygun bir yöntem olduğu gösterilmiştir.

Bilim Kodu : 905.1.034
Anahtar Kelimeler : Mikroşerit antenler, Faz dizisi, Genetik algoritma, Işıma örüntüsü
Sayfa Adedi : 100
Tez Yöneticisi : Doç. Dr. Erkan AFACAN

OPTIMIZATION OF PHASED ARRAY MICROSTRIP ANTENNAS USING GENETIC ALGORITHM

(M.Sc. Thesis)

Mustafa Selim CAN

GAZİ UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

July 2010

ABSTRACT

Nowadays, rise in the number of signal sources coupled with increasing electromagnetic noises in the atmosphere, prevents having signals in a desired quality. For this reason, antenna arrays are frequently used in telecommunication systems. Because antenna arrays, in contrast to single antenna elements, enables desired radiation pattern by changing related parameters so that they are preferred in industrial applications. It is an important field in communications provided that the antenna arrays are effectively process signals coming from a proposed direction with maximum gain, as well as they blocks signals coming from undesired directions.

In this study, antenna arrays designed with circular microstrip antennas are taken into account and they are subject to optimization to obtain desired radiation patterns. Genetic algorithms which was emerged from genome science and provides a framework for complex problems having no optimal solution with analytic methods, are used in this study as an optimization method. Consequently, it is proved that genetic algorithm is an optimum way for having best parameters showing main beam in antennas in the right direction.

Science Code : **905.1.034**
Key Words : **Microstrip antennas, Phased array, Genetic algorithm, Radiation pattern**
Page Number : **100**
Adviser : **Assoc. Prof. Dr. Erkan AFACAN**

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi birikimi ve katkılarıyla beni yönlendiren ve yardımlarını esirgemeyen tez danışmanım Sayın Doç. Dr. Erkan AFACAN'a teşekkürü bir borç bilirim.

Tezin yazımı aşamasında yardımcı olan değerli arkadaşlarım Mesut Çeviker, Hüseyin Gören ve Mustafa Öner Dikdere'ye teşekkür ederim.

Son olarak; eğitim hayatım boyunca her zaman yanımda olan ve beni destekleyen, maddi ve manevi yardımlarını hiç bir zaman eksik etmeyen çok değerli aileme sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
E	Elektrik alan şiddeti
H	Manyetik alan şiddeti
a	Anten yarıçapı
a_e	Anten efektif yarıçapı
L	Anten uzunluğu
h	Dielektrik taban kalınlığı
f_r	Rezonans frekansı
λ_0	Boşluktaki dalgaboyu
λ_d	Dielektrik tabandaki dalgaboyu
η_o	Serbest uzay dalga empedansı
k_o	Serbest uzay dalga sayısı
ϵ_r	Bağıl dielektrik sabiti

Kısaltmalar	**Açıklama**
GA	Genetik algoritma
EDD	Eşdüzlemsel dalga

1. GİRİŞ

Elektromanyetik sinyalleri göndermek ve almak için kullanılan anten elemanları, fiziksel özellikleri nedeniyle her yöne eşit olarak ışıma yapmamaktadırlar. Antenin atmosferde ya da boşlukta yaptığı ışımanın belirli bölgelerde daha fazla yapılmasının istendiği durumlarda anten dizileri kullanılmaktadır. Anten dizileri, radar ve kablosuz haberleşme sistemlerinde yoğun olarak kullanılmakta olup; demet tarama, demet şekillendirme, yüksek yönelticilik ve kazanç gerektiren uygulamalarda başarı sağlamaktadırlar.

Anten dizileri, genellikle özdeş olan belirli sayıda anten elemanının bir araya getirilmesiyle oluşturulur. Anten dizileri doğrusal, düzlemsel ve hacimsel dizilime göre çeşitlendirilirler. Anten elemanlarının ışıma örüntüsünden farklı olarak anten dizilerinin ışıma örüntüleri, dizi faktörü olarak adlandırılan fonksiyon vasıtasıyla elde edilir. Anten dizisinin parametrelerinin değişimi dizi faktörünü etkiler, bu sayede ışıma örüntüsü şekillendirilir. Genel anten incelemelerinde, seçilen antenin ışıma örüntüsü, empedansı, ana hüzme genişliği ve bant genişliği gibi temel parametreleri literatürde varolan yöntemlerle hesaplanır. Ancak anten dizisi incelemelerinde, istenen ışıma örüntüsünü en iyi yaklaşıklıkla verecek akım genliği, faz uyarımı veya elemanları arasındaki uzaklıkları parametreleri belirlenmeye çalışılır.

Işıma örüntüsü incelemeleri genel olarak üç grupta sınıflandırılır. İlk grupta, önceden belirlenmiş doğrultularda sıfırlara sahip ışıma örüntüsünü veren çalışmalar yer alır. Bu gruba Schelkunoff yöntemi örnek verilebilir [1]. Bu metodda analitik yaklaşımla istenilen doğrultularda sıfırlara sahip bir örüntü oluşturmak amaçlanır. Ancak bu metod sıfırlanan doğrultular haricindeki bölgelerde kontrole sahip değildir. İkinci grupta, tüm görünen bölgede istenen ışıma örüntüsünü yaklaşık olarak verebilecek bir anten tasarımı yer alır. Fourier dönüşüm metodu [2], Woodward-Lawson metodu [3], ve Orchard-Elliot metodu [4], bu grup için verilebilecek örneklerdendir. Üçüncü grupta ise, dar ana hüzmeye ve düşük yan hüzmelere sahip ışıma örüntüleri

sağlayan anten dizisi tasarımı yer almaktadır. Dolph-Chebyshev [5] ve Taylor metodu [6], üçüncü grup için örnek olarak verilebilir.

Anten dizisi incelemelerinde, genetik algoritma etkin olarak kullanılan optimizasyon metodlarından biridir. Genetik algoritmalar problemler için kesin sonuç vermemekle birlikte, çözüm için en iyi yaklaşımı verirler. Genetik algoritma, genetik biliminden esinlenerek ortaya atılmıştır ve genetik biliminde yer alan mutasyon, çaprazlama gibi uygulamaları algoritma dahilinde uyarlayarak oluşturulan popülasyonun türevlerinden en iyi sonucu bulmayı hedefler. Analitik metodlarda ve klasik optimizasyon yöntemlerinde problemin elde edilen tek bir çözümü olmasına karşın, genetik algoritma çözümlerinde esnek ve genel çözümler üretilebilmektedir. Bu yöntemlere nazaran daha yavaş olsa da, gelişen bilgisayar teknolojileri sayesinde bu dezavantajın etkisi azalmış ve geniş bir uygulama alanı bulmuştur.

Bu tez çalışmasında, düzlemsel anten dizisinin elemanlarının faz parametreleri değiştirilerek önceden belirlenmiş doğrultuda ana hüzmeye sahip ışıma örüntüsü elde etmek amacıyla genetik algoritma metodu kullanılmıştır. Çalışmada özdeş dairesel mikroşerit antenlerden oluşan bir anten dizisi modellenmiş, anten dizisinin akım genliği ve elemanları arasındaki uzaklık değerleri sabit tutularak faz değerleri optimize edilmiş ve başarılı sonuçlar elde edilmiştir.

Tez çalışmasının ikinci bölümünde mikroşerit antenlerden bahsedilmiş, bu antenlerin yapıları ve çeşitleri anlatılmıştır. Çalışmada kullanılan mikroşerit dairesel antenlerin üzerinde ayrıca durulmuştur.

Üçüncü bölümde, anten dizilerinin yapıları, çeşitleri ve uygulamaları hakkında bilgi verilmiştir. Ayrıca, anten dizilerinin dizi faktörü için genel formülasyonlar sunulmuştur.

Dördüncü bölümde, tez çalışmasında optimizasyon yöntemi olarak kullanılan genetik algoritma tanıtılmıştır. Genetik algoritmanın yapısı, avantajları, dezavantajları ve kullandığı işlem metodları sunulmuştur.

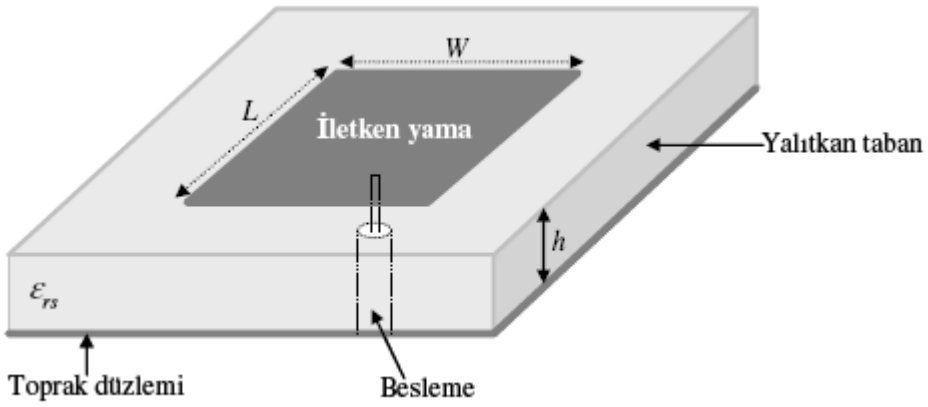
Beşinci bölümde, dairesel mikroşerit anten dizisinin optimizasyonu için sunulan çözüm yaklaşımı tanıtılmıştır. Anten dizisinin modellenmesi ve genetik algoritma ile optimizasyonu için kullanılan yöntemler adımlarıyla birlikte tanıtılmıştır.

Son bölümde, elde edilen sonuçlar değerlendirilmiş ve daha sonra yapılabilecek çalışmalar için fikirlere yer verilmiştir.

2. MİKROŞERİT ANTENLER

Yüksek performans gerektiren havacılık, uzay ve askeri alanlardaki haberleşme uygulamalarında da; ağırlık, büyüklük, maliyet, kolay kurulum gibi gereksinimlere karşılık veren antenlere ihtiyaç duyulmaktadır. Bu ihtiyaçlara karşılık verebilmek için mikroşerit antenler kullanılır. Bu antenler, modern baskı devre teknolojisiyle kolayca üretilebilmesi, monte edildiği yüzeyde mekanik olarak sağlamlık güvencesi vermesi, verildiği geometrik şekle göre istenilen rezonans frekansı, polarizasyon, ışıma örüntüsü ve empedans özelliklerinin kolayca elde edilebilmesi nedenleriyle tercih edilmektedir [2].

Mikroşerit anten fikrinin ortaya atılması ve patentinin alınması 1950'li yıllara dayanmasına rağmen, ancak 1970'li yıllarda dikkat çekmeye başlamıştır [7,8]. Bir mikroşerit anten basit olarak, çok ince metal bir şeritin ($t << \lambda_0$) bir dielektrik malzeme ($h << \lambda_0$, genelde $0.003\lambda_0 \leq h \leq 0.005\lambda_0$) üzerine yerleştirilmesiyle elde edilir. Mikroşerit yama, maksimum örüntüsü yamanın normaline gelecek şekilde tasarlanır. Bu da, yamanın altındaki doğru uyarım modunun seçilmesiyle elde edilir. Örneğin bir dikdörtgen yama için, yamanın genişliği L, genelde $\lambda_0/3 \leq L \leq \lambda_0/2$ olacak şekilde alınır. Yama ve toprak yüzeyi bir dielektrik tabaka ile ayrılır ve alt katman olarak adlandırılır.



Şekil 2.1. Mikroşerit anten yapısı [2]

Mikroşerit anten yapısı Şekil 2.1'de gösterilmektedir. Mikroşerit antenlerin tasarımında çok çeşitli alt katmanlar kullanılabilir. Bu alt katmanların dielektrik sabiti $2.2 \leq \epsilon_r \leq 12$ aralığındadır. Anten performansının daha iyi olması için, bu aralığın alt sınırına en yakın dielektrik sabite sahip kalın alt katmanlar istenir, çünkü bu tür alt katmanlar daha verimli, geniş band aralığında, boşlukta yayılmak için gevşek bağıl alana sahip olma özelliklerini sunarlar. Ancak bu durumun antenin büyüklüğüyle beraber maliyetini de artırma dezavantajı vardır [9]. İnce alt katmanlar ise; istenmeyen yayılım ve kuplajların engellenmesi için sıkı bağıl alanların bulunduğu mikrodalga devrelerinde kullanılır ve böylece daha küçük anten yapısı elde edilir. Fakat büyük kayıplara sahip olmaları nedeniyle, verimlilikleri daha azdır ve bant genişlikleri daha dardır. Mikroşerit antenler genellikle mikrodalga devrelerine entegre edilmelerinden ötürü, iyi bir anten performansı ve devre tasarımının elde edilmesi için parametrelerin uygun bir biçimde seçilmesi gerekir.

2.1. Mikroşerit Antenlerin Avantajları ve Dezavantajları

Sahip olduğu özellikler sayesinde 100 MHz'den 50 GHz'e kadar geniş bir frekans aralığında kullanılabilen mikroşerit antenlerin, diğer antenlere göre bazı üstünlükleri ve dezavantajları bulunmaktadır [10].

Avantajları:

1) Hafif ve küçük hacimli olması,
2) Düşük üretim maliyeti,
3) Düzlemsel biçimliliği nedeniyle kullanışlı olması,
4) Çok ince yapılı olmaları nedeniyle uzay ve hava araçlarının aerodinamik yapısını bozmamaları,
5) Güdümlü mermiler, roketler ve uyduların üzerine önemli değişikliğe neden olmadan monte edilebilmeleri,
6) Düşük saçılma kesit alanına (scattering cross-section) sahip olmaları
7) Besleme konumunda yapılan ufak değişikliklerle doğrusal ve dairesel ışıma yapabilmeleri,

8) Boşluk desteği gerektirmemeleri,
9) Osilatörler, yükselteçler, değişken zayıflatıcılar, anahtarlar, modülatörler, karıştırıcılar, faz değiştiricileri vs. katıhal araçlarının, mikroşerit antenlerin alt taşına ilave edilerek bileşik sistemler meydana getirilebilmesi,
10) Besleyici hatları ve uyumlandırma devrelerinin, mikroşerit antenle birlikte eşzamanlı üretilebilir biçimde olması.

Dezavantajları:

1) Dar band genişliği,
2) Çeşitli kayıplar sonucu, düşük kazançlı olması,
3) Mikroşerit antenlerin çoğunun yarı düzlem içinde ışıması,
4) -20 dB olan en üst kazancın elde edilmesinde pratik zorlukların olması,
5) Düşük boyuna (end-fire) ışıma performansı,
6) Besleyici ışıma elemanı arasındaki zayıf yalıtım,
7) Yüzey dalgaları uyarımının mümkün olabilmesi,
8) Düşük güç kapasitesine sahip olması,

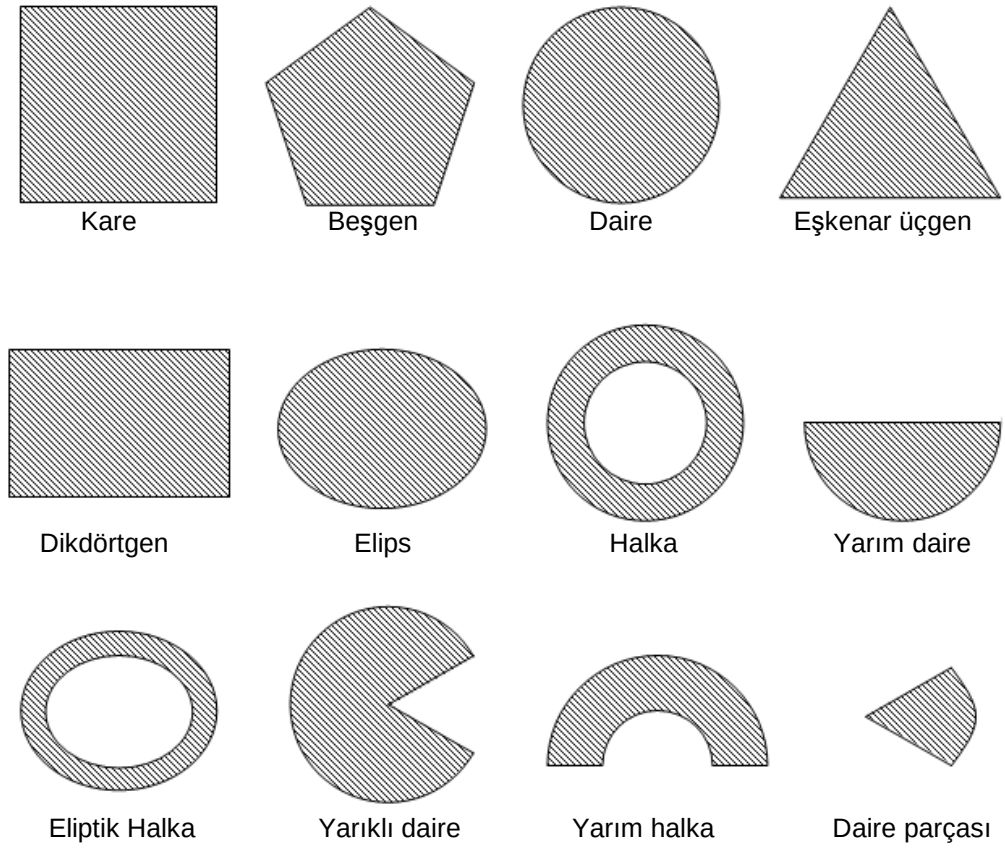
2.2. Mikroşerit Anten Çeşitleri

Mikroşerit antenler, üretiminin kolay olması nedeniyle çok çeşitli yapılarda tasarlanabilirler. Bu da mikroşerit antenlerin geniş bir yelpazede sınıflandırılabilmesine neden olmuştur. Tez çalışmasında, dairesel mikroşerit yama antenler kullanılmıştır. Yama antenler; dikdörtgen, dairesel, üçgen, halka gibi farklı geometrik şekillerde tasarlanabilmektedir. Mikroşerit yama antenlerin yanısıra; monopol, dipol, boşluk, yürüyen dalga, halka gibi çok çeşitli yapılar da mevcuttur.

2.2.1. Mikroşerit yama antenler

Mikroşerit yama antenler, dielektrik malzemenin üzerine baskı devre metoduyla herhangi bir geometrik şeklin iletken bir yama olarak işlenmesiyle oluşturulur. Bazı yama anten şekilleri Şekil 2.2’de gösterilmiştir. Birçok yama

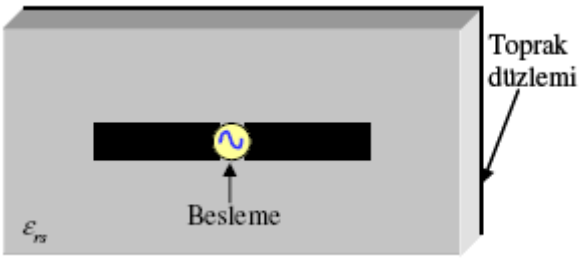
anten için ışıma karakteristiği incelenmiş durumdadır. Bu tip antenlerde en çok tercih edilen şekil, dikdörtgen ve dairedir. Tipik olarak kazançları 5-6 dB seviyesindedir ve 70^o-90^o arasında 3-dB hüzme genişliğine sahiptirler.



Şekil 2.2. Mikroşerit Yama Anten Çeşitleri [10]

2.2.2. Baskılı dipol antenler

Baskılı dipoller, uzunluk-genişlik oranları nedeniyle dikdörtgen yama antenlerden farklılık arz ederler. Bir dipolün genişliği tipik olarak 0.05 λ_0'dan daha azdır. Dipol ve yama antenin ışıma örüntüleri benzer boylamsal akım dağılımı nedeniyle benzerlikler içermektedir. Ancak, ışıma dirençleri ve çapraz polarizasyon ışımaları birbirinden farklıdır. Dipol antenler, küçük boyutları ve lineer polarizasyona sahip olmaları nedeniyle tercih edilirler. Mikroşerit dipol anten örneği şekil 2.3'de gösterilmiştir.

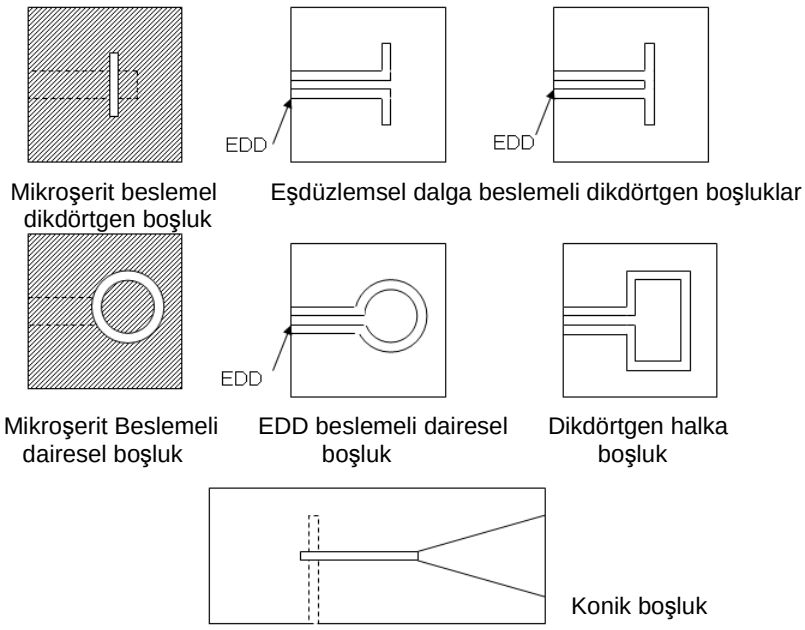


Şekil 2.3. Mikroşerit Dipol Anten [11]

Alt katmanın kalın olduğu mikroşerit dipol antenler, bu sayede iyi bir bant aralığı elde edilmesini sağlayan yüksek frekans değerleri için uygundur.

2.2.3. Mikroşerit boşluk antenler

Mikroşerit boşluk antenler, dielektrik alt katman üzerine yerleştirilmiş metal bir yüzeyin üzerinde geometrik bir boşluk oluşturulmasıyla elde edilir. Bu geometrik boşluk, herhangi bir şekilde olabilir. Mikroşerit boşluk anten örnekleri Şekil 2.4'de gösterilmiştir.

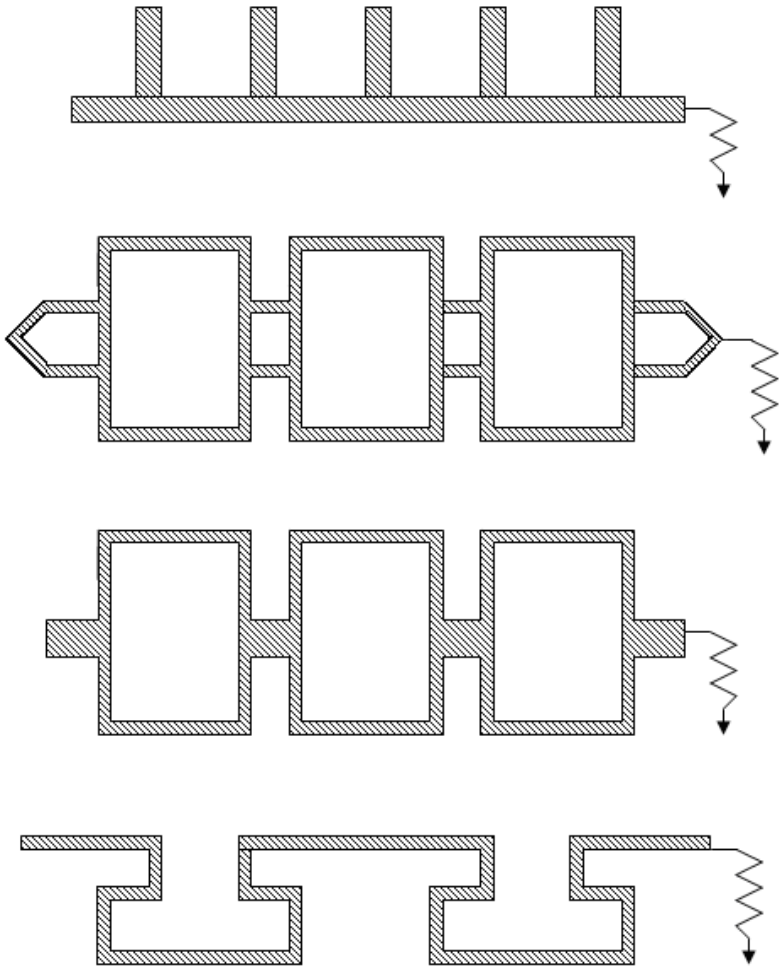


Şekil 2.4. Mikroşerit Boşluk Anten [12]

Teorik olarak, boşluk antenin şekli birçok mikroşerit yama anten şeklinde olabilir, ancak literatürde bu şekillerin sadece belli başlı olanlarının karakteristikleri incelenmiştir. Yama antenler gibi, boşluk antenler de mikroşerit hat ya da eşdüzlemsel dalga kılavuzu ile beslenir.

2.2.4. Mikroşerit yürüyen dalga antenler

Mikroşerit yürüyen dalga antenler; yeterli genişliği sahip olarak zincir şeklinde periyodik olarak sıralanmış uzun bir mikroşerit hattan meydana gelir. Antenin diğer tarafı, duran dalga oluşmasını engellemek için dengeli rezistif yük ile sonlandırılır. Bu anten türünün bir örneği Şekil 2.5'de gösterilmiştir.



Şekil 2.5. Mikroşerit yürüyen dalga anten çeşitleri [12]

Yürüyen dalga antenler, yatay ve düşey konum arasında istenilen açıda yayılım yapabilecek şekilde tasarlanabilirler.

Mikroşerit yama anten, mikroşerit boşluk anten ve baskılı dipol antenin karakteristik özellikleri Çizelge 2.1'de ele alınmıştır.

Çizelge 2.1 Mikroşerit Anten Çeşitlerinin Karşılaştırılması [12]

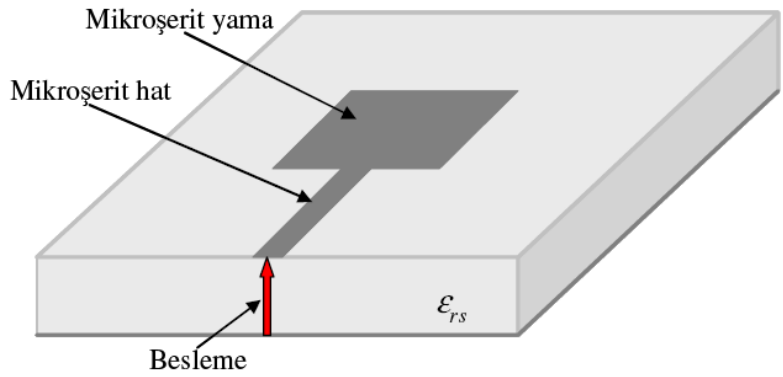
Karakteristik	**Mikroşerit Yama Anten**	**Mikroşerit Boşluk Anten**	**Baskılı Dipol Anten**
Profil	İnce	İnce	İnce
Üretim	Çok kolay	Kolay	Kolay
Polarizasyon	Lineer ve dairesel	Lineer ve dairesel	Lineer
Çift Frekansta Çalışabilme	Mümkün	Mümkün	Mümkün
Şekil Esnekliği	Her şekilde olabilir	Genelde dikdörtgen ve dairesel	Genelde dikdörtgen ve dairesel
Band Genişliği	%2-%50	%5-%30	~%30

2.3. Mikroşerit Anten Besleme Şekilleri

Mikroşerit antenler, dielektrik katmanın tek bir tarafından yayıldıkları için, ilk olarak yalnızca mikroşerit bir hat ile ya da toprak katmanından çekilen koaksiyel prob ile besleniyordu. Daha sonraları geliştirilen tekniklerle, mikroşerit antenlerin yakın kuplajlı besleme ve boşluk kuplajlı besleme yoluyla da beslenebilmesi sağlanmıştır.

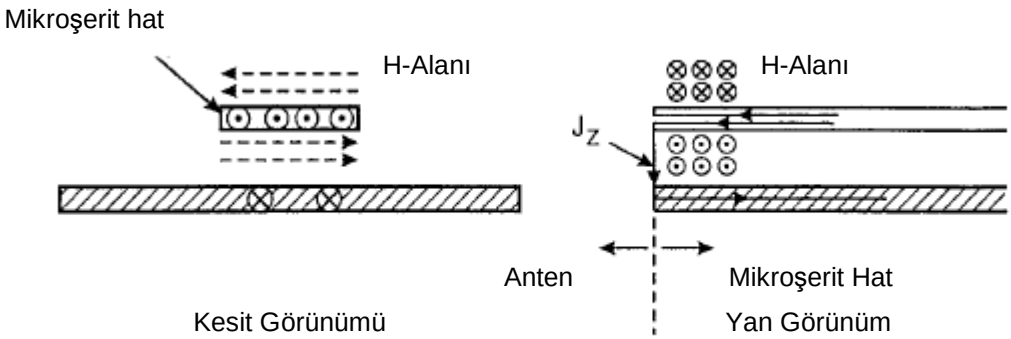
2.3.1. Mikroşerit besleme

Eşyüzeyli besleme olarak da bilinen mikroşerit besleme, Şekil 2.6'da gösterildiği gibi, yama antenle aynı düzlemde yer alan bir besleme şeridinin antene eklenmesiyle yapılır. Yamanın bir parçası kabul edilebilmesi ve fabrikasyon sırasında ikisinin de eş zamanlı üretilebilmesi nedeniyle en kolay besleme biçimi gibi görünmektedir. Fakat mikroşerit besleme tekniğinin bazı dezavantajları vardır.



Şekil 2.6 Mikroşerit hatlı besleme [11]

Şekil 2.7'de, mikroşerit besleme yoluyla beslenen bir antenin birleşme düzlemindeki uyarımı eşdeğer J_z elektrik akım yoğunluğu ve H_y manyetik alanı kullanılarak gösterilmiştir. Şekilde noktalı çizgiler H, düz çizgiler akım çizgilerini göstermektedir.

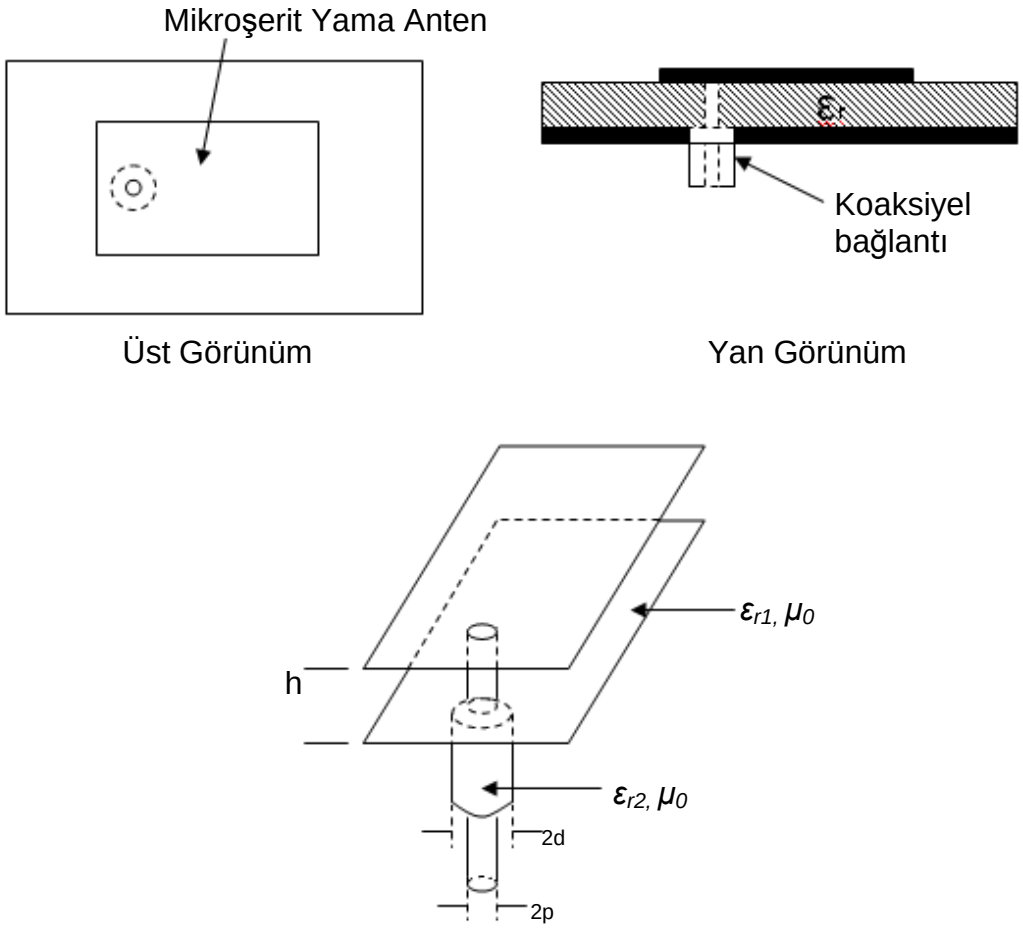


Şekil 2.7. Yama antenin ve besleme şeridinin arasındaki H_y'nin eşdeğer akım yoğunluğu J_z ile gösterimi [12]

Mikroşerit besleme yapılan antenlerde, dielektrik katman kalınlığı arttıkça yüzey dalgaları ve parazitik besleme ışıması artar, bu da pratik uygulamalarda band genişliğinin %2 - %5 oranında azalmasına neden olur [2]. Ayrıca, kalın dielektrik katmanlar için giriş empedansı oldukça endüktiftir, bu da zayıf bir empedans uyumuna neden olur. Mikroşerit beslemede empedans uyumu, besleme noktasının değiştirilmesiyle sağlanır [13].

2.3.2. Koaksiyel besleme

Koaksiyel Besleme tekniğinde, Şekil 2.8'de görüldüğü gibi, anteni bir boşluktan besleyen koaksiyel hattın iç iletkeni antene güç iletirken, dış iletkeni toprak yüzeyine bağlanır. Koaksiyel hattın yama üzerindeki konumu değiştirilerek istenen en iyi empedans uyumu sağlanabilir.

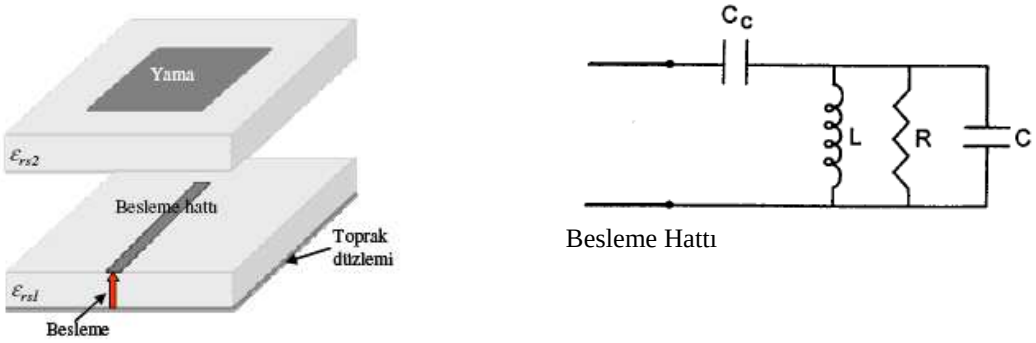


Şekil 2.8. Koaksiyel besleme [12]

Bir yama anten koaksiyel besleme yoluyla toprak yüzeyinin altından beslendiğinde, besleme probu alt katman kalınlığıyla doğru orantılı bir seri reaktansa neden olur. Bu durum kalın alt katmanlar için anten ışıma rezistansını önemli ölçüde etkiler ve uygun empedans eşlemesini zorlaştırır [14]. Ayrıca yapısı gereği koaksiyel besleme, diğer besleme tekniklerine göre üretimi bir hayli zorlaştırmaktadır.

2.3.3. Yakınlık kuplajlı besleme

Elektromanyetik kuplajlı besleme olarak da bilinen yakınlık kuplajlı besleme, Şekil 2.9'da görüldüğü gibi birbirine temas etmeyen farklı düzlemlerdeki anten ve besleme hattından oluşur. Bu besleme tekniğinde iki farklı alt katman kullanılır. Besleme hattı iki katmanın arasına yerleştirilerek yama antenin altında açık olarak sonlandırılır. Doğal olarak hat ile yama arasındaki kuplajlanma kapasitiftir.



(a) Fiziksel görünüm (b) Eşdeğer devre şeması

Şekil 2.9. Yakınlık Kuplajlı Besleme [11]

Şekil 2.9'da bu besleme şeklinin eşdeğer devresi verilmiştir. Burada kuplajlanma kapasitörü C_c, yama anteni temsil eden RLC rezonans devresine seri bağlıdır. Hattın açık ucu sivri bir şekilde sonlandırılarak ve iki alt katmanın parametreleri uygun bir şekilde seçilerek band genişliği artırılabilir. %13 değerinin üzerinde bant genişliği elde edebilmek için bu besleme tekniği kullanılır [15].

2.3.4. Açıklık kuplajlı besleme

Bu besleme tekniğinin bir örneği Şekil 2.10'da gösterilmiştir. Açıklık kuplajlı beslemenin avantajları, daha geniş band aralığı sağlaması ve besleme hattından çıkan ışımanın, yayılım yapan yamadan yalıtılabilmesidir. Açıklık kuplajlı beslemede, boşluk kuplajlı beslemede olduğu gibi iki farklı alt katman kullanılır ve toprak yüzeyi bu iki katman arasına yerleştirilir. Besleme hattı ile yama arasındaki elektromanyetik kuplajlama, toprak yüzeyinde oluşturulan bir boşluk yordamıyla yapılır. Bu boşluk herhangi bir biçimde olabilir. Bu boşluğun geometrik parametreleri değiştirilerek band genişliğinin iyileştirilmesi sağlanır. Katmanların dielektrik sabitleri ve kalınlıkları ışıma özelliklerini etkileyen unsurlardandır. Örneğin besleme hattının üzerindeki katmanın ince ve yüksek dielektrik sabite, yamanın altındaki katmanın kalın ve düşük dielektrik sabite sahip olması gerekir. Ayrıca, besleme hattının açık ucu ile yama arasında toprak yüzeyi sayesinde ekranlama sağlandığından, polarizasyon saflığı sağlanır [12].

Çizelge 2.2'de mikroşerit antenlerin farklı besleme şekillerinin karşılaştırması yapılmıştır [12].

Çizelge 2.2. Mikroşerit antenlerin farklı besleme şekillerinin karşılaştıması

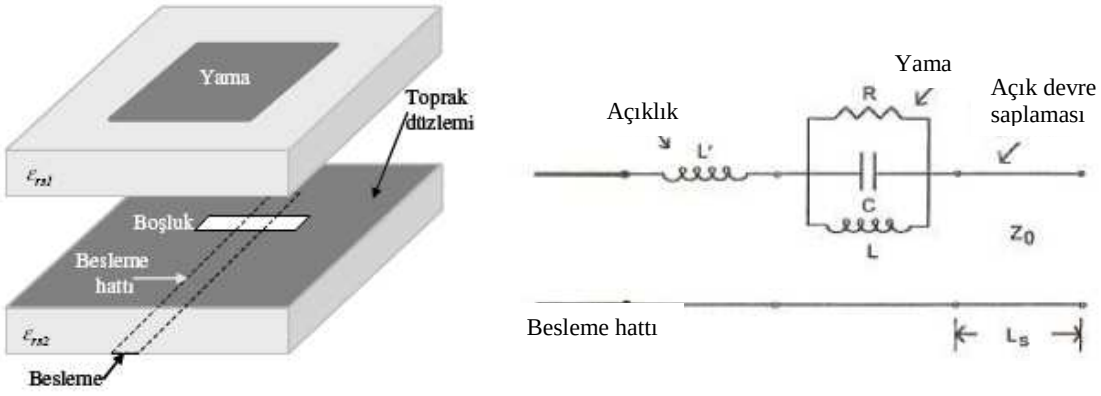
Karakteristik	**Mikroşerit Besleme**	**Koaksiyel Besleme**	**Yakınlık Kuplajlı**	**Açıklık Kuplajlı**
Yapı	Eşdüzlemli	Farklı düzlemli	Paralel düzlemli	Paralel düzlemli
Sahte besleme yayılımı	Çok	Az	Az	Yok
Üretim kolaylığı	Kolay	Delme ve lehimleme gerekli	Paralel hizalama gerekli	Paralel hizalama gerekli
Güvenilirlik	Çok iyi	Lehimleme nedeniyle az	İyi	İyi

Çizelge 2.2. (Devam) Mikroşerit antenlerin farklı besleme şekillerinin karşılaştıması

Empedans uyumu	Kolay	Kolay	Kolay	Kolay
Band genişliği (empedans uyumunda)	%2-5	%2-5	%13 [16]	%21 [17]

2.4. Dairesel Mikroşerit Antenler

Dairesel mikroşerit antenler, dikdörtgen antenlerden sonra en çok kullanılan antenlerden biridir. Temel bir mikroşerit dairesel anten biçimi Şekil 2.11'de gösterilmektedir. Performans olarak dikdörtgen antenlere benzerler, ve geometrik olarak biraz daha küçüktürler. Anten dizileri gibi bazı yaygın uygulamalarda, dairesel mikroşerit antenlerin belirgin üstünlükleri vardır. Çeşitli empedans değerleri, ışıma örüntüleri ve çalışma frekansları oluşturabilmek için kolayca modifiye edilebilirler. Dikdörtgen antenlerde modların sırası yamanın uzunluğu ve genişliği ile değiştirilebilirken, dairesel antenlerde yalnızca dairenin yarıçapı etkendir. Bu durum modların sırasını değiştirmez, fakat her birinin rezonans frekansının mutlak değerini değiştirir [18].



(a) Fiziksel görünüm (b) Eşdeğer devre şeması

Şekil 2.10. Açıklık kuplajlı besleme [11]

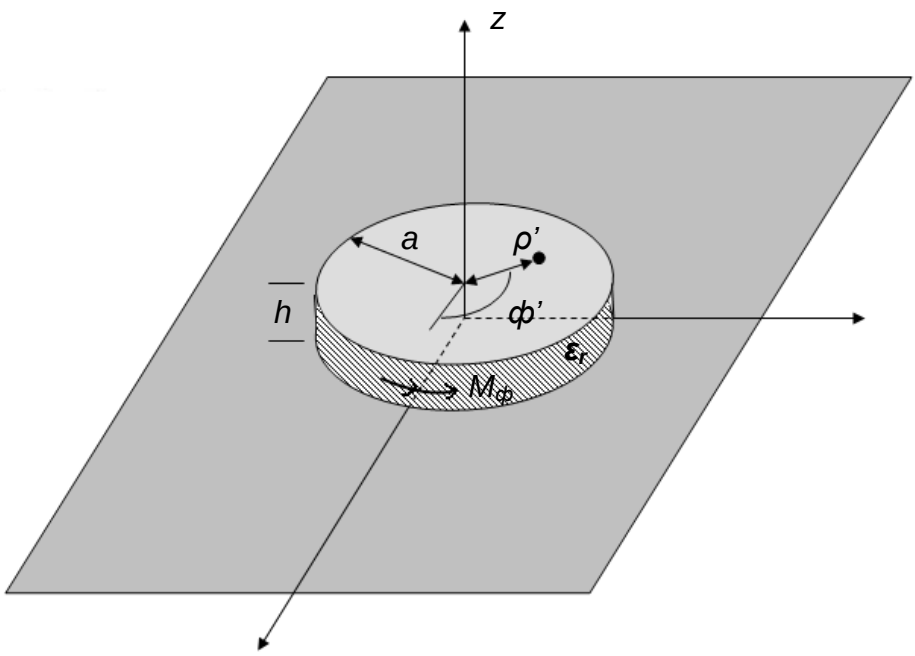
Tam dalga modelinin aksine, dairesel mikroşerit antenler kavite modeli kullanılarak analiz edilir. Kavite modelinde, yamanın daire şeklinde olması nedeniyle silindirik koordinatlar kullanılır. Bu modelde, yamayı ve toprak yüzeyini temsil edecek şekilde üstte ve altta iki mükemmel elektrik iletken ve dairesel kavitenin etrafını saran manyetik bir iletken düşünülür. Alt katmanın dielektrik malzemesinin yamanın kenarlarından itibaren kesik olduğu düşünülür.

2.4.1. Kavite modeli yoluyla elektrik ve manyetik alan yayılımının elde edilmesi

Dairesel mikroşerit anten, dielektrik alt katman üzerine yerleştirilen ince bir dairesel iletkenden oluşur. Dielektrik katmanın kalınlığı olan h, λ_0'dan çok küçük olduğundan, antenin yalnızca z yönünde elektrik alanı vardır, ve manyetik alanın yalnızca ρ ve φ bileşenleri bulunmaktadır. Şekil 2.11'de dairesel yamaya ait kavite modeli ifade edilmiştir. Mikroşerit diskin kenarlarının normalindeki akım bileşeni kenarlarda sıfıra yaklaşır. Bu da diskin kenarlarındaki manyetik alanın tanjantının sıfır olmasına neden olur. Bu varsayımlarla birlikte, mikroşerit disk, üstü ve altı elektrik bir duvarla, kenarları da manyetik bir duvarla kapanmış silindirik bir kavite modeli olarak ele alınabilir. Bu nedenle, TM_{nm} moduna karşılık gelen dielektrik bölgedeki alanlar, bir kavite için tanımlanmış dalga eşitliği kullanılarak çözülebilir [2].

Bir çok tipik mikroşerit anten için h çok küçük olduğundan ($h < 0.05\lambda_0$), z yönündeki alanlar sabittir. Bu nedenle TM^z_{mn} modunda kavitenin ve dolayısıyla mikroşerit antenin rezonans frekansı, şu denklemle elde edilir:

$$f_{r_{mn()}} = \frac{1}{2\pi\sqrt{\mu \in}}\left(\frac{X'_{mn}}{a}\right) \qquad (2.1)$$



Şekil 2.11. Dairesel mikroşerit anten yapısı ve kavite modeli [2]

X'_{mn} ifadesi, $J_m(x)$ Bessel fonksiyonunun türevinin sıfırlarını ifade eder. X'_{mn} için ilk üç değer sırasıyla şu şekildedir:

$$X'_{21} = 3.0542 \quad (2.2a)$$

$$X'_{11} = 1.8412 \quad (2.2b)$$

$$X'_{21} = 3.0542 \quad (2.2c)$$

$$X'_{31} = 4.2012 \quad (2.2d)$$

Dominant mod, TM^z_{110} modudur ve bu moda ait rezonans frekansı şu şekildedir:

$$(f_r)_{110} = \frac{1.8412}{2\pi\sqrt{\mu\in}} = \frac{1.8412 v_0}{2\pi a\sqrt{\in_r}} \quad (2.3)$$

(2.3) eşitliğinde ifade edilen rezonans frekansı antenin kenarlarını dikkate almaz. Kenar kısımlar da dikkate alındığında anten elektriksel olarak daha

büyük görünür, bu nedenle gerçek *a* yarıçapı, Eş. 2.4 eşitliğinde ifade edilen ve a_e olarak tanımlanan efektif yarıçap ile değiştirilir.

$$a_e = a\left\{1+\frac{2h}{\pi a \in_r}\left[\ln\left(\frac{\pi a}{2h}\right)+1.7726\right]\right\}^{1/2} \tag{2.4}$$

Hiç bir uyarım akımının bulunmadığı durumda mikroşerit antenin elektrik alanı için dalga eşitliği şu şekilde yazılabilir:

$$(\nabla^2 + k^2)\vec{E} = 0 \qquad k = 2\pi\sqrt{\in_r}\,/\,\lambda_0 \tag{2.5}$$

Kavitenin içindeki elektrik alan yukarıdaki dalga eşitliğini ve manyetik duvar bağıl koşullarını sağlamalıdır. Silindirik koordinatlardaki dalga eşitliği şu çözümle sağlanır:

$$E_z = E_0 J_1(k\rho')\cos\phi' \tag{2.6}$$

Burada $J_1(k\rho)$ birinci derecede Bessel fonksiyonunu işaret eder. *E*'nin yalnızca z bileşeninin olması ve $\partial / \partial z = 0$ olması nedeniyle, manyetik bileşenler şu hale gelir:

$$H_\rho = j\frac{E_0}{\omega\mu_0}J_1'(k\rho')\sin\phi' \tag{2.7a}$$

$$H_\rho = j\frac{E_0}{\omega\mu_0}J_1'(k\rho')\cos\phi' \tag{2.7b}$$

Bu iki denklemden ilk $k\rho$'ye göre türevi ifade eder. Kavitenin içindeki diğer alan ifadeleri sıfırdır. Yani:

$$E_\rho = E_\phi = H_z = 0 \tag{2.8}$$

olur. Burada ’ işareti *∂/ ∂ρ* türevini ve *Φ*’ ifadesi dairesel antenin çevresindeki azimut açısını ifade etmektedir.

(2.6) eşitliğinden yola çıkarak, manyetik akım yoğunluğu şu şekilde yazılabilir:

$$M_s = -2\hat{n} \times E_\alpha \,|_{\rho'=a_e} = \hat{a}_\phi 2E_0 J_1(ka_e)\cos\phi' \qquad E_\alpha = E_z \tag{2.9}$$

Alt katmanın yüksekliğinin çok az olması ve (2.9)’da belirtilen akım yoğunluğunun z doğrultusunda tekdüze olması nedeniyle, iplik biçimindeki manyetik yoğunluğa şu şekilde yaklaşımda bulunulabilir:

$$I_m = hM_s = \hat{a}_\phi 2hE_0 J_1(ka_e)\cos\phi' = \hat{a}_\phi 2V_0 \cos\phi' \tag{2.10}$$

Bu denklemde,

$$V_0 = hE_0 J_1(ka_e) \text{ ve } \phi' = 0 \tag{2.11}$$

şeklindedir.

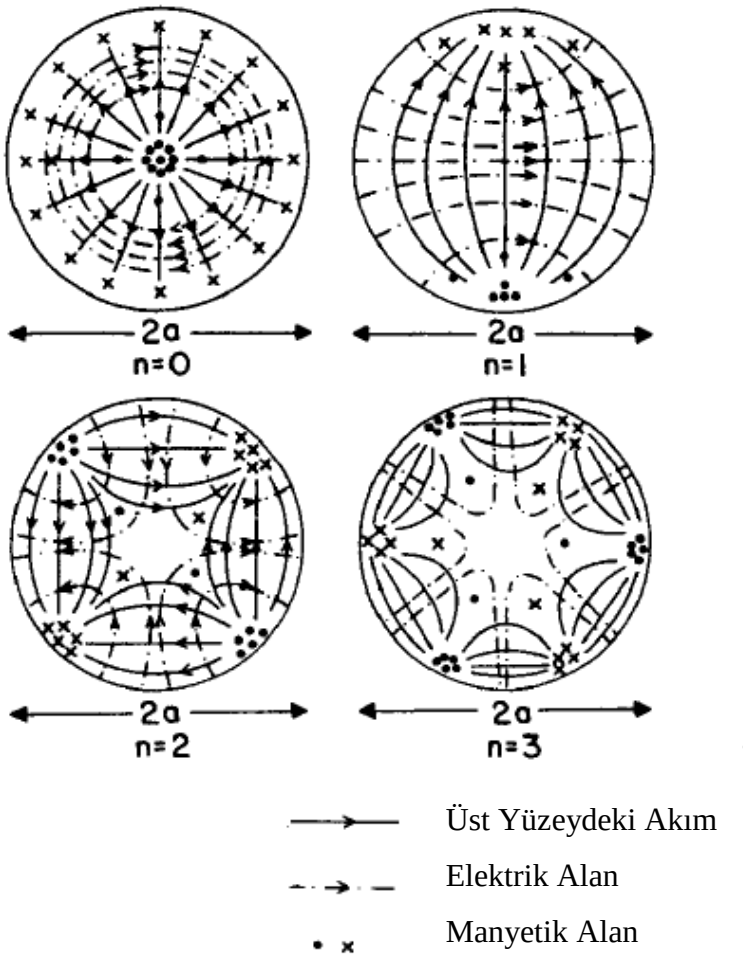
TM_{01}, TM_{11}, TM_{21}, TM_{31} modlarında rezonans durumundaki alan ve yüzey akımı örüntüleri Şekil 2.12’de betimlenmiştir [19].

Işınım eşitlikleri kullanılarak, elektrik alan ifadeleri şu şekilde yazılabilir:

$$E_r = 0 \tag{2.12a}$$

$$E_\theta = -j\frac{k_0 a_e V_0 e^{-jk_0 r}}{r}\{\cos\phi J'_{02}\} \tag{2.12b}$$

$$E_\phi = -j\frac{k_0 a_e V_0 e^{-jk_0 r}}{2r}\{\cos\phi \sin\phi J_{02}\} \tag{2.12c}$$



Şekil 2.12. Rezonans durumunda çeşitli modlarda alan ve yüzey akımı örüntüleri [19]

Bu denklemlerdeki J'_{02} ve J_{02} ifadeleri Bessel fonksiyonları cinsinden şu şekilde ifade edilir:

$$J'_{02} = J_0(k_0 a_e \sin\theta) - J_2(k_0 a_e \sin\theta) \qquad (2.13a)$$

$$J_{02} = J_0(k_0 a_e \sin\theta) + J_2(k_0 a_e \sin\theta) \qquad (2.13b)$$

Denklemlerde yer alan a_e ise, Eş. 2.4 denkleminde ifade edilen efektif yarıçaptır. Esas yüzeylerdeki alan ifadeleri şu şekilde kısaltılabilir:

E-düzlemi ($\Phi = 0^0, 180^0, 0^0 \leq \theta \leq 90^0$)

$$E_\theta = j\frac{k_0 a_e V_0 e^{-jk_0 r}}{2r}\left[J'_{02}\right] \quad (2.14a)$$

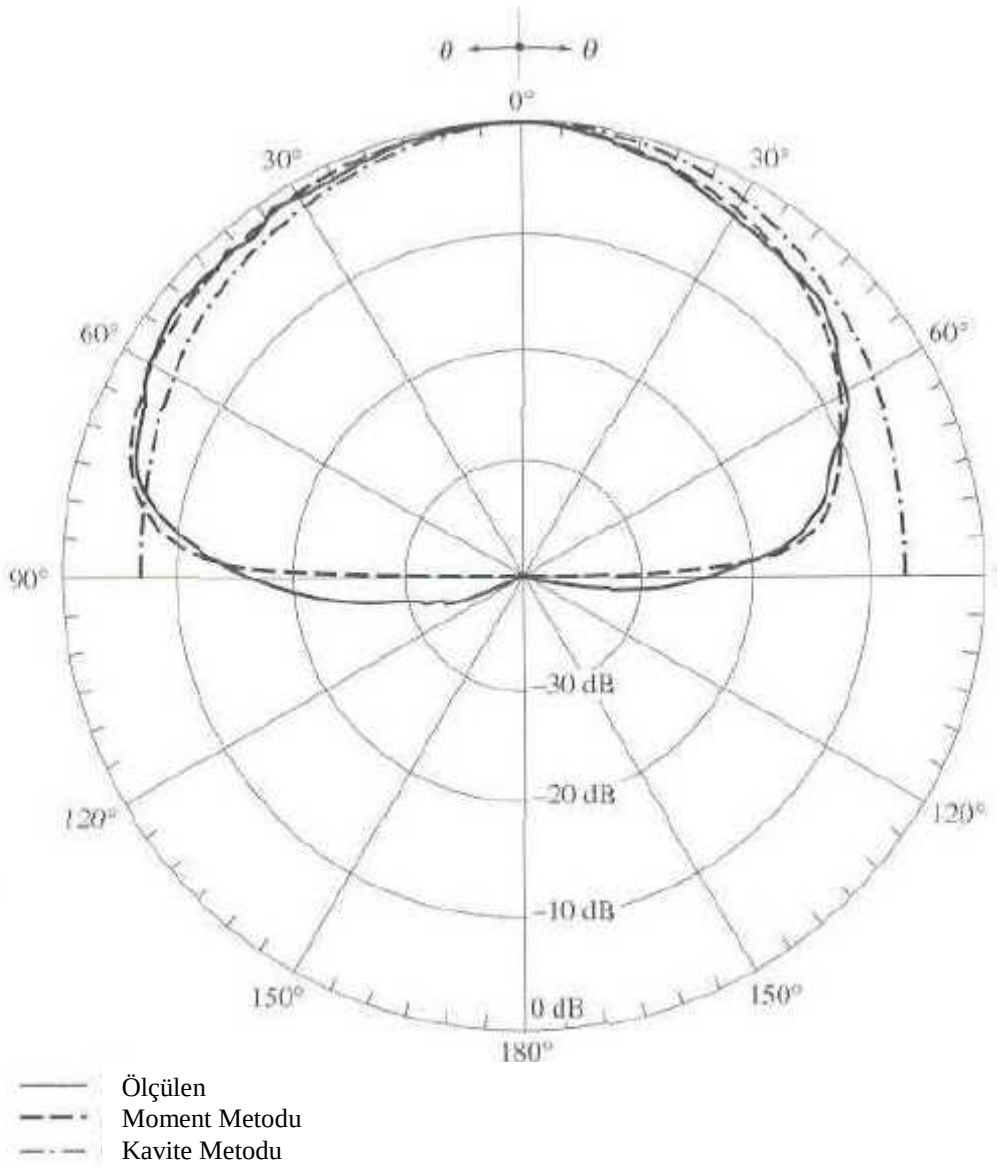
$$E_\phi = 0 \quad (2.14b)$$

H-düzlemi ($\Phi = 90^0, 270^0, 0^0 \leq \theta \leq 90^0$)

$$E_\theta = 0 \quad (2.15a)$$

$$E_\phi = j\frac{k_0 a_e V_0 e^{-jk_0 r}}{2r}\left[\cos\theta\, J_{02}\right] \quad (2.15b)$$

Şekil 2.13'de, Eş. 2.10a ve Eş.2.10b kullanılarak kavite modeli ve moment metodu yoluyla hesaplanan ve ölçülen ışıma örüntüleri yer almaktadır [2]. Ölçülen ve hesaplanan örüntüler arasında asimetrinin bulunması, ölçüm sırasında besleme hattının E-düzlemi boyunca simetrik olarak yerleştirilmemesinden kaynaklanmaktadır. Moment metodu analizinde besleme hattının konumu önemli iken, kavite modelinde besleme hattının konumunun önemi yoktur. Mikroşerit antenin alt yüzeyi toprak katmanı ile kaplandığından anten alt tarafa ışıma yapmamaktadır. Teorik ölçümlerde ışıma örüntüsünün alt kısmının gösterilmemesinin nedeni budur. Yapılan ölçümlerde de antenin alt yüzeye ışıma yapmadığı görülebilmektedir. Bunun yanı sıra mikroşerit anten yan yüzeylere de ışıma yapmamakta ve 90 derece doğrultusunda anten ışıması sönümlenmektedir. Moment metodunun, kavite modeline göre daha doğru sonuçlar verdiği ölçülen değerler dikkate alındığında görülebilir. Kavite modelinde, hesaplamaların kolaylaştırılması amacıyla bazı varsayımların yapılması gerçek sonuçlardan biraz sapılmasına neden olabilir. Ancak teorik hesaplamaların yapılmasında kavite modelinin kolaylık sağlaması optimum sonucun elde edilebilmesi için tercih nedeni olmaktadır.



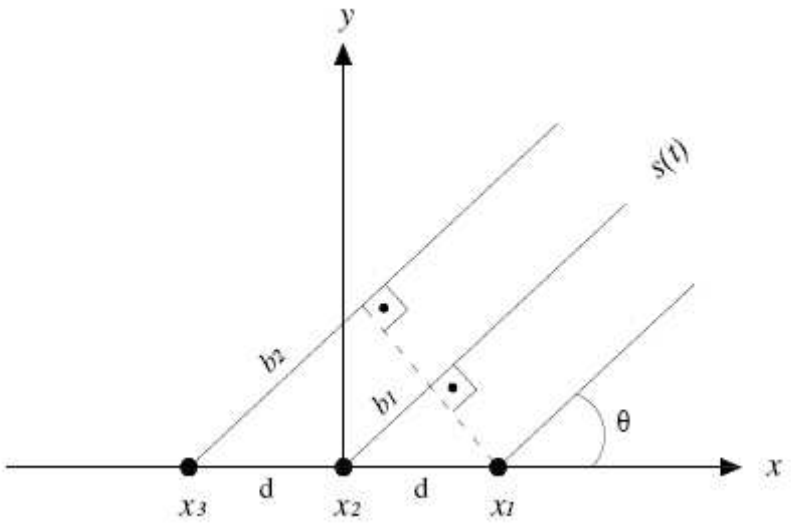
Şekil 2.13. Dairesel mikroşerit anten için moment metodu ve kavite metodu kullanılarak hesaplanan ve ölçülen E-düzlemi ışıma örüntüsü (a=0.525 cm, a_e = 0.598 cm, ρ_f = 0.1 cm, ϵ_r = 2.2, h = 0.1588 cm, f_0 = 10 GHz, λ_0 = 3 cm) [2]

3. ANTEN DİZİLERİ

Antenlerin bir çok türü mevcuttur. Her anten türünün kendine has özellikleri olmasına karşın, bazı durumlarda tek başına bir anten istenilen gereksinimleri karşılayamamaktadır. Tek bir anten genelde düşük kazanç sağlar ve hüzme genişliği tek bir yöne doğru yoğunlaşır. Bazı uygulamalarda, kazancın artırılması ve hüzme genişliğinin istenilen yöne doğru yoğunlaştırılması istenir. Bu durumda birden fazla antenin geometrik olarak yerleştirilmesiyle oluşturulan anten dizileri kullanılır. Anten dizileri herhangi bir anten türü seçilerek oluşturulabilir. Anten dizisindeki her bir eleman farklı yapıda seçilebilir, ancak dizinin toplam alanını ve örüntüsünü hesap etmek için özdeş elemanlar tercih edilir. Birden fazla antenin geometrik olarak üretimini ve tasarımını kolaylaştırması nedeniyle mikroşerit antenler, en çok kullanılan anten türlerinden biridir [20].

Dizi antenlerin elektrik ve manyetik alanları, her bir antenden ışıyan alanların vektörel olarak toplanmasıyla elde edilir. İstenen yöne doğru bir ışıma örüntüsü elde edebilmek için, her bir antenden ışıyan alanın istenilen doğrultuda birbirine eklenmesi, istenmeyen doğrultularda ise birbirini yoketmesi sağlanmalıdır. Anten dizilerinde, dizinin toplam ışıma örüntüsünü şekillendirebilmek için beş kontrol elemanı vardır. Bunlar:

1. Anten dizisinin geometrik yerleşimi (lineer, dairesel, küresel vb.),
2. Dizi elemanları arasındaki uzaklık,
3. Her bir elemanın uyarım genliği
4. Her bir elemanın uyarım fazı
5. Her bir elemanın ışıma örüntüsü



Şekil 3.1. İki boyutlu uzayda sinyalin zaman gecikmesi ile yönlendirilmesi

En basit anten dizisi yapısı, iki antenin yanyana getirilmesiyle elde edilen iki elemanlı anten dizisidir. İki elemanlı anten dizisi, dizi antenlerin yapısını analiz etmek için uygun bir tercihtir.

3.1. İki Elemanlı Anten Dizisi

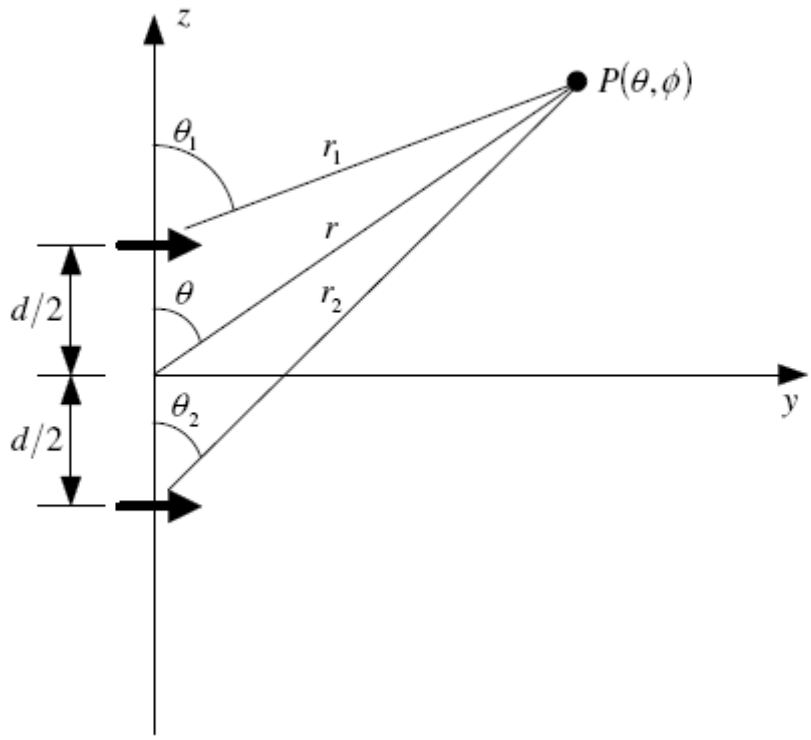
İki elemanlı anten dizisi, Şekil 3.2.'de görüleceği gibi iki anten elemanının z düzleminde yatay olarak yanyana yerleştirilmesiyle elde edilmiştir.

Tek bir elemanın uzak alandaki elektrik ifadesi:

$$E = \hat{a}_\theta \, j\eta \frac{kI_0 l}{4\pi r} e^{-jkr} \cos(\theta), \quad kr >> 1 \tag{3.1}$$

şeklinde yazılabilir. İki eleman arasında kuplajlanmanın olmadığı düşünülerek, y-z düzleminde yaydıkları toplam elektrik alanı şu şekilde verilebilir:

$$E_t = E_1 + E_2 = \hat{a}_\theta \, j\eta \frac{kI_0 l}{4\pi} \left\{ \frac{e^{-j[kr_1 - (\beta/2)]}}{r_1} \cos\theta_1 + \frac{e^{-j[kr_2 + (\beta/2)]}}{r_2} \cos\theta_2 \right\} \tag{3.2}$$



Şekil 3.2. İki elemanlı anten dizisi

Burada η boşluktaki empedansı (120π), k faz sabitini (2π/λ), I_0 uyarım akımı genliğini, l elemanlar arasındaki uzaklığı, β elemanlar arasındaki uyarım akımının faz farkını temsil etmektedir. Elektrik alan ifadesi uzak alan için oluşturulduğundan θ_1, θ_2 ve θ değerlerinin birbirine eşit olduğu varsayılabilir. Bu durumda faz değişkenleri için

$$r_1 \approx r - \frac{d}{2}\cos\theta \tag{3.3a}$$

$$r_2 \approx r + \frac{d}{2}\cos\theta \tag{3.3b}$$

ve genlik değişkenleri için

$$r_1 \approx r_2 \approx r \tag{3.4}$$

kabul edilebilir. Bu varsayımlar dikkate alınarak Eş. 3.2 yeniden düzenlenirse:

$$E_T = E_1 + E_2 = \hat{a}_\theta \, j\eta \frac{kI_0 l e^{-jkr}}{4\pi r} \left|\cos\theta\right| \left[e^{+j(kd\cos\theta+\beta)/2} + e^{-j(kd\cos\theta+\beta)/2}\right] \quad (3.5)$$

$$E_T = \hat{a}_\theta \, j\eta \frac{kI_0 l e^{-jkr}}{4\pi r} \cos\theta \left[\frac{1}{2}(kd\cos\theta + \beta)\right] \quad (3.6)$$

Eş. 3.1 ve Eş. 3.6 karşılaştırıldığında; dizinin toplam alanının, tek bir elemanın alanının dizi faktörü (AF) olarak adlandırılan bir çarpan ile çarpılarak elde edildiği açıkça görülmektedir. Sabit genliklere sahip iki elemanlı anten dizisi için dizi faktörü:

$$AF = 2\cos\left[\frac{1}{2}(kd\cos\theta + \beta)\right] \quad (3.7)$$

olarak verilir. n elemanlı bir dizi için AF şu şekilde normalize edilir:

$$(AF)_n = \cos\left[\frac{1}{2}(kd\cos\theta + \beta)\right] \quad (3.8)$$

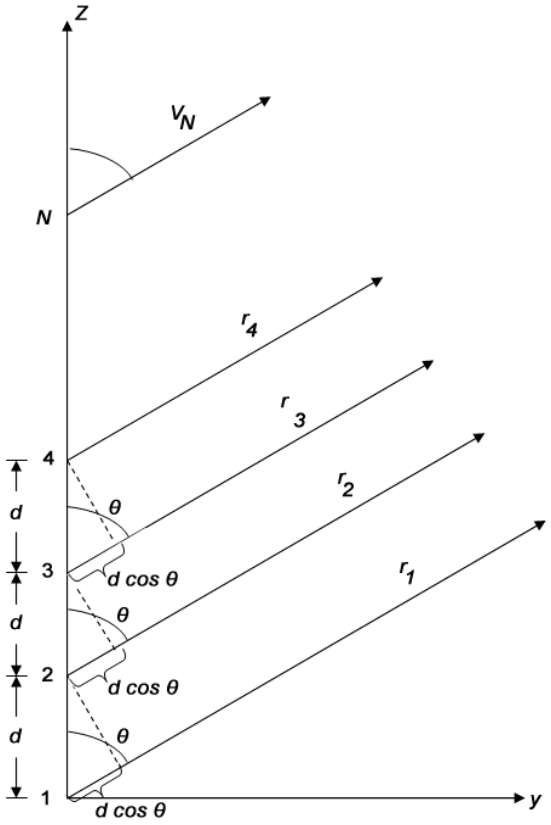
AF, dizinin geometrisinin ve uyarım fazının bir fonksiyonudur. Elemanlar arasındaki uzaklıklar (d) ve faz (β) değiştirilerek, dizi çarpanının ve dolayısıyla dizinin alanlarının karakteristikleri değiştirilebilir. Bu durum, ikiden fazla elemana sahip anten dizileri için de geçerlidir. Kısaca,

E(toplam) = [E(tek bir eleman) x [AF(dizi çarpanı)] (3.9)

olarak ifade edilir.

3.2. N Elemanlı Doğrusal Anten Dizisi

İki elemanlı anten dizisi gerçekte bir tür doğrusal anten dizisidir. Ancak anten dizisinin karakteristiğini daha iyi kontrol edebilmek için, eleman sayısının artırılması gerekir. N elemanlı doğrusal anten dizisi, Şekil 3.3.'de görüldüğü gibi, N adet anten elemanının tek bir doğrultu üzerinde yerleştirilmesiyle elde edilir.



Şekil 3.3. Z ekseni üzerine yerleştirilmiş özdeş elemanlardan oluşan doğrusal anten dizisi [2]

Doğrusal anten dizisinin her bir elemanının özdeş ve uyarım akımı faz farklarının bir yanındakine göre β kadar farkının olduğunu düşünürsek, dizi çarpanı AF Eş. 3.10 ve Eş. 3.11 eşitliklerinde olduğu gibi ifade edilir:

$$E_T = a_1 e^{j(1/2)kd\sin\theta} + a_2 e^{j(3/2)kd\sin\theta} + \ldots + a_N e^{j[(2N-1)/2]kd\sin\theta} \quad (3.10)$$

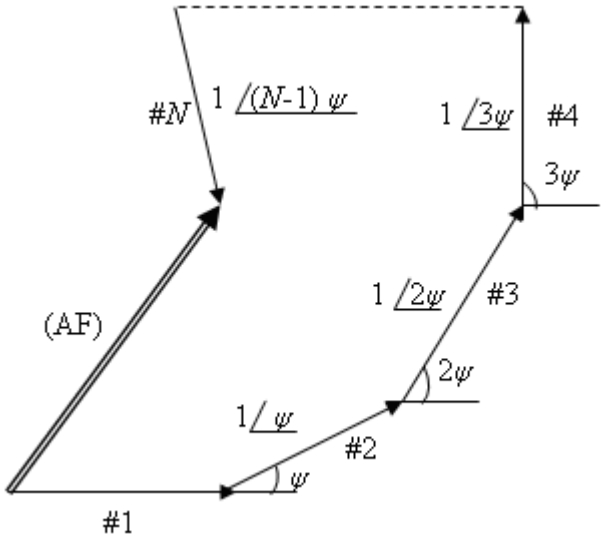
$$E_T = \sum_{n=1}^{N} a_n e^{j[(2n-1)/2]kd\sin\theta} \tag{3.11}$$

kd cos *θ* + *β* ifadesini *ψ* sembolüyle kısaltacak olursak,

$$AF = \sum_{n=1}^{N} e^{j(n-1)\psi} \tag{3.12}$$

şeklinde yazılır. Özdeş elemanlardan oluşan bir dizi için toplam dizi faktörü eksponansiyel çarpanların toplamıdır. Bu durumda dizi çarpanı, her bir elemanın birim genliğinin ve bir önceki elemana göre artan *ψ* faz değerini ifade eden N fazörünün toplamı olarak düşünülebilir. Bu durum Şekil 3.4'de grafiksel olarak betimlenmiştir.

Şekil 3.4.'de, dizinin elemanları arasındaki *ψ* bağıl faz değerinin değiştirilerek dizi faktörünün fazının ve genliğinin değiştirilebildiği görülmektedir.



Şekil 3.4. Z ekseninde dizilmiş özdeş kaynaklardan oluşan N elemanlı dizinin fazör diyagramı [2]

Eş. 3.12 eşitliğinin her iki yanı $e^{j\psi}$ ile çarpıldığında,

$$(AF)e^{j\psi} = e^{j\psi} + e^{j2\psi} + e^{j3\psi} + ... + e^{j(N-1)\psi} + e^{jN\psi} \tag{3.13}$$

eşitliği elde edilir. Eş. 3.13 Eş 3.12'den çıkarılırsa,

$$AF = \left[\frac{e^{jN\psi} - 1}{e^{j\psi} - 1}\right] = e^{j[(N-1)/2]\psi}\left[\frac{e^{j(N/2)\psi} - e^{-j(N/2)\psi}}{e^{j(1/2)\psi} - e^{-j(1/2)\psi}}\right] \tag{3.14}$$

$$AF = e^{j[(N-1)/2]\psi}\left[\frac{\sin\left(\frac{N}{2}\psi\right)}{\sin\left(\frac{1}{2}\psi\right)}\right] \tag{3.15}$$

elde edilir. Dizinin referans noktası, merkez noktası olarak seçilirse, Eş. 3.15 ile verilen dizi faktörü şu şekilde yazılır:

$$AF = \left[\frac{\sin\left(\frac{N}{2}\psi\right)}{\sin\left(\frac{1}{2}\psi\right)}\right] \tag{3.16}$$

Dizi çarpanının maksimum değerinin birim değer olması için normalizasyon işlemi gerçekleştirilir. Bu durumda dizi faktörü,

$$(AF)_n = \frac{1}{N}\left[\frac{\sin\left(\frac{N}{2}\psi\right)}{\sin\left(\frac{1}{2}\psi\right)}\right] \tag{3.17}$$

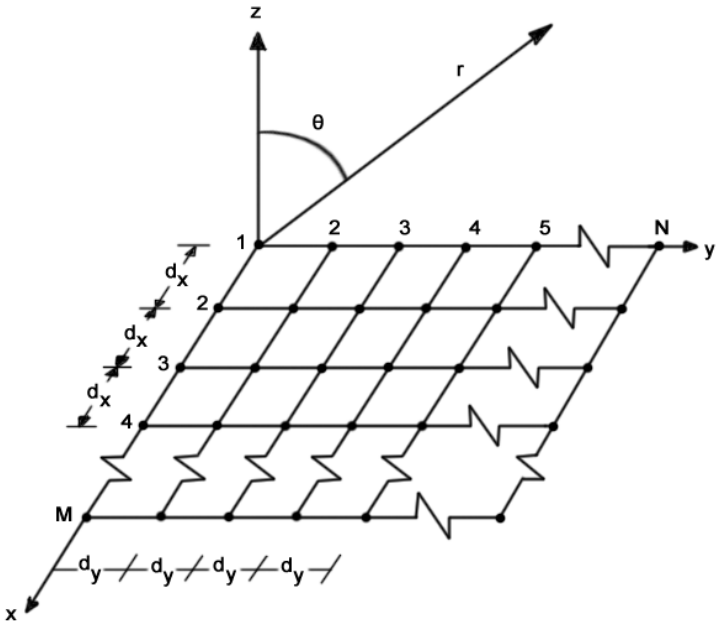
olur. ψ değerinin çok küçük olduğu durumlarda ise,

$$(AF)_n \cong \left[\frac{\sin\left(\frac{N}{2}\psi\right)}{\sin\left(\frac{1}{2}\psi\right)}\right] \tag{3.18}$$

şeklinde ifade edilir.

3.3. Düzlemsel Anten Dizisi

Anten dizileri, yalnızca bir doğru üzerine elemanların yerleştirilmesinin yanısıra, Şekil 3.5.'de görüleceği gibi bir düzlem üzerine yerleştirilerek de oluşturulur [2]. Bu şekilde oluşturulan düzlemsel anten dizilerinin belirgin avantajları vardır. Düzlemsel anten dizileri, daha düşük kenar loblarına sahip ve daha simetrik ışıma örüntüleri oluşturulmasını sağlar. Ayrıca, doğrusal anten dizilerinden farklı olarak yalnızca bir düzlem üzerinde değil, uzayın herhangi bir noktasına ana ışıma hüzmesinin doğrultulmasını sağlar. Bu tür antenler, radar sistemleri başta olmak üzere birçok haberleşme uygulamasında yaygın olarak kullanılmaktadır.



Şekil 3.5. Düzlemsel Anten Dizisi [2]

Düzlemsel antenlerin dizi çarpanını elde etmek için, anten dizisinin N adet anten elemanından meydana gelen M adet doğrusal anten dizisinin yanyana getirilmesi ile oluşturulduğu düşünülebilir. Doğrusal anten dizisi için Eş. 3.12 ile verilen dizi faktörü ifadesini kullanacak olursak,

$$AF = \sum_{m=1}^{M} I_{m1} e^{j(m-1)(kd_x \sin\theta\cos\phi+\beta_{x})} \tag{3.19}$$

elde edilir. Bu ifadede I_{m1} her bir doğrusal dizi için uyarım akımını ifade etmektedir. Bu diziler arasındaki uzaklık d_x, bir öncekine göre faz farkı ise β_x olarak ele alınırsa, tüm düzlemsel yüzey için dizi faktörü:

$$AF = \sum_{n=1}^{M} I_{1n} \left[\sum_{m=1}^{M} I_{m1} e^{j(m-1)(kd_x \sin\theta\cos\phi+\beta_x)} \right] e^{j(n-1)(kd_y \sin\theta\sin\phi+\beta_y)} \tag{3.20}$$

olur. Tüm dizi için uyarım akımının aynı olduğu varsayılırsa:

$$AF = I_0 \sum_{m=1}^{M} e^{j(m-1)(kd_y \sin\theta\cos\phi+\beta_y)} \sum_{n=1}^{N} e^{j(n-1)(kd_y \sin\theta\sin\phi+\beta_y)} \tag{3.21}$$

ifadesi elde edilir. İki elemanlı anten dizisinde yapılan normalizasyon işlemini düzlemsel anten için de yapacak olursak,

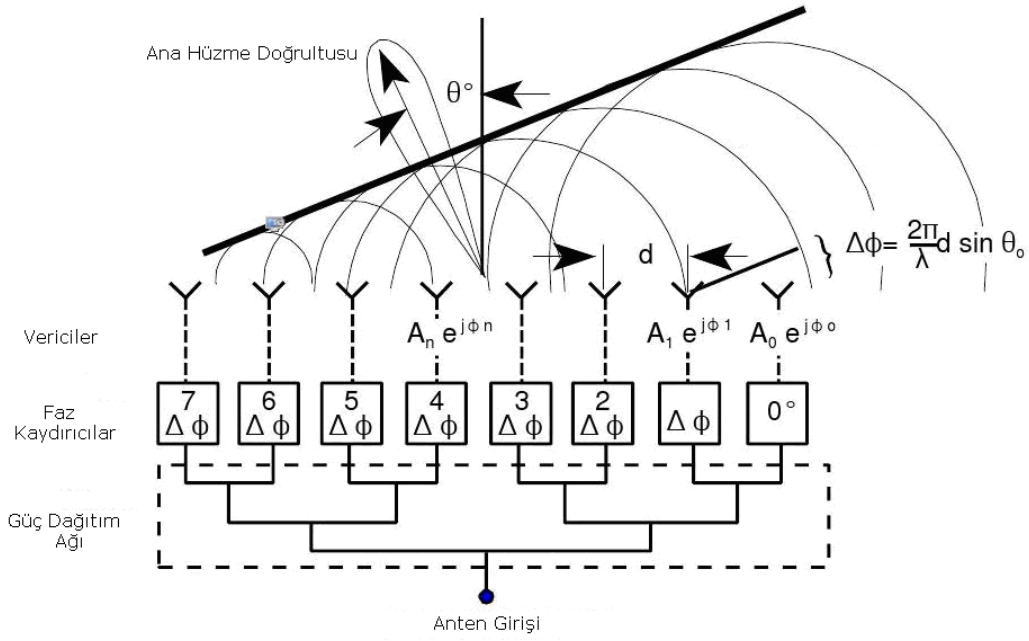
$$AF_n(\theta,\phi) = \left\{ \frac{1}{M} \frac{\sin\left(\frac{M}{2}\psi_x\right)}{\sin\left(\frac{\psi_x}{2}\right)} \right\} \left\{ \frac{1}{N} \frac{\sin\left(\frac{N}{2}\psi_y\right)}{\sin\left(\frac{\psi_y}{2}\right)} \right\} \tag{3.22}$$

öyle ki:

$$\psi_x = kd_x \sin\theta\cos\phi + \beta_x \tag{3.23a}$$

$$\psi_y = kd_x \sin\theta \sin\phi + \beta_y \tag{3.23b}$$

ifadesi elde edilir. Bu kısaltılmış ifade, elemanlar arasındaki faz farkının bir öncekine eşit olduğu durum için oluşturulmuştur. Elemanlar arasındaki faz farkının bir öncekine eşit olması durumu, pratikte antenlerin beslenmesini kolaylaştırmaktadır. Şekil 3.6.'da bir anten dizisinin pratikte nasıl beslendiği gösterilmektedir.



Şekil 3.6. Bir anten dizisinin beslenme şekli

Anten dizileri arasındaki faz farkının sabit olması yerine, her bir anten elemanının uyarım fazının farklı olması da istenebilir. Bu durumda ışıma örüntüsünün daha esnek kontrolü sağlanır. Bu durumda anten dizisinin dizi faktörü şu şekilde verilebilir [21]:

$$AF(\theta,\phi) = \sum_{i=1}^{N} \alpha_i e^{j\beta_i} e^{j2\pi dx_i \sin\theta\cos\phi} e^{j2\pi dy_i \sin\theta\sin\phi} \tag{3.24}$$

Bu eşitlikte N toplam anten elemanı sayısını, θ z-eksenine göre yükselme açısını, φ x-eksenine göre azimut açısını, a_i i'nci elemanın uyarım akımı genliğini, β_i i'inci elemanın akımını göstermektedir. dx_i ve dx_y ise anten elemanının sırasıyla y eksenine ve x eksenine uzaklıklarını göstermektedir. Bu eşitlik kullanılarak, birbirinden farklı fazlara ve eksenlerden farklı uzaklıklara sahip anten elemanlarından oluşan düzlemsel bir dizinin dizi çarpanı elde edilebilmektedir.

4. GENETİK ALGORİTMA

Genetik algoritma, genetik biliminin ve doğal seleksiyon teorisinin ortaya attığı prensipleri temel alarak oluşturulmuş bir optimizasyon tekniğidir. Optimizasyon, en basit tabiriyle bir işlemin sonuçlarını en iyi hale getirmeye çalışmaktır. Bir problemin tek bir sonucu değil de birden fazla sonucu var ise, sonuçlar arasından en iyisini elde etmek için optimizasyon teknikleri kullanılır. Genetik algoritma, bir çok bireyden oluşan bir popülasyonun belirlenen seçim kuralları sonucu evrilerek en uygun bireyin elde edilmesini sağlar. Bu metod ilk olarak 1975 yılında John Holland tarafından geliştirilmiş [22], ardından doğalgaz iletim hatlarıyla ilgili zor bir problem üzerinde doktora tezi hazırlayan öğrencisi David Goldberg tarafından yaygınlaştırılmıştır [23].

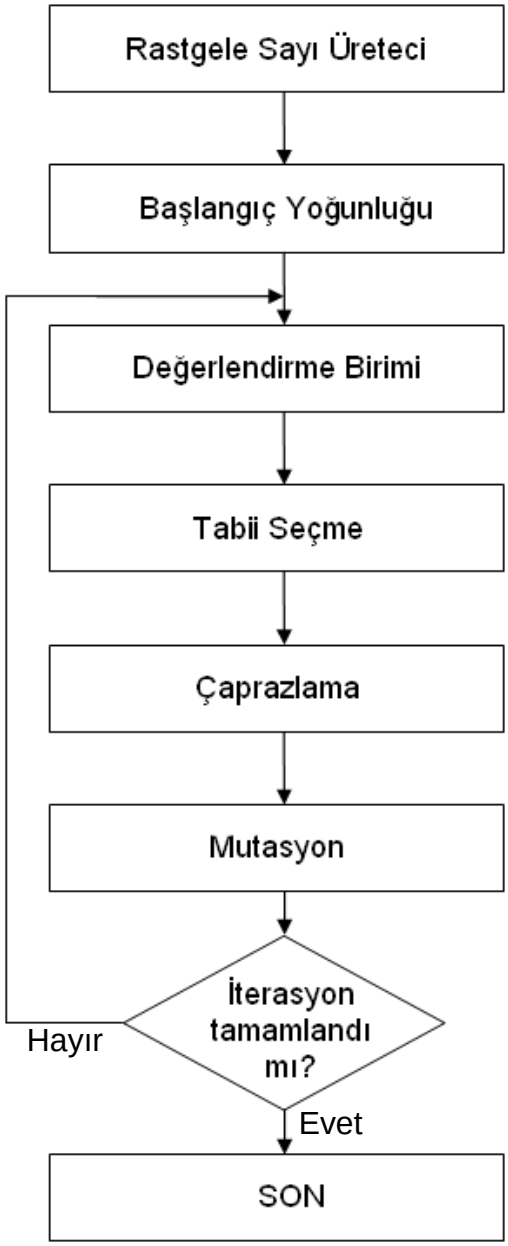
Genetik algoritmanın avantajları şu şekilde sıralanabilir [24]:

1) Sürekli ve ayrık değişkenler üzerine uygulanabilmesi
2) Problemin doğasıyla ilgili detaylı bilgiye ihtiyaç duymaması
3) Geniş bir örnekleme alanında eşzamanlı arama yapabilmesi
4) Büyük sayılarda değişkenlerle çalışabilmesi
5) Paralel bilgisayarlar için uygun olması
6) Sadece bir sonucu değil, birden fazla optimum sonucu liste halinde sunabilmesi
7) Sayısal olarak üretilmiş verilerle, deneysel verilerle ve analitik fonksiyonlarla çalışabilmesi.

4.1. Genetik Algoritmanın Yapısı

Genetik algoritma, çözümlenmesi hedeflenen problemin ögelerinden bir başlangıç popülasyonu oluşturulması ile başlar. Başlangıç popülasyonu oluşturulduktan sonra; sırasıyla üreme, çaprazlama ve mutasyon operatörleri kullanılarak bir sonraki kuşaktaki çözüm üretilir. Tekrar üreme işlemi boyunca, seçme işlemini gerçekleştirmek için her bir bireye uygunluk

fonksiyonu uygulanır. Optimal bir çözümün bulunmasına dek, birbirini takip eden kuşakların gelişmesi ve değerlendirmesi döngüsel olarak devam eder. Şekil 4.1'de tipik bir genetik algoritma yapısının akış diyagramı verimiştir.



Şekil 4.1. Genetik Algoritma Akış Diyagramı

Genetik algoritmanın, seçilen problemin çözümündeki başarı kriteri, genetik algoritma işlemlerinin uygulanmasının ardından elde edilen ve problemin çözümünü temsil eden bireylerin gösterimidir. Popülasyondaki bireylerin

problemin çözümü olup olmadığına karar vermek için uygunluk fonksiyonunun işlevi önemlidir. Bireye uygulanan uygunluk fonksiyonundan dönen değer yüksek ise, bireyin popülasyondaki diğer bireyler ile çoğalmasına izin verilir. Bu bireylerin çaprazlanması sonucu çocuk adı verilen yeni bireyler üretilir. Çocuk, kendisini meydana getiren bireylerin özelliklerini taşır. Uygunluk fonksiyonu sonrası düşük değer elde edilen bireylerin seçilme şansı az olduğundan, bu bireyler döngü devam ettikçe popülasyon dışında kalacaklardır. Yeni popülasyon, bir önceki popülasyonda uygunluk değeri yüksek bireylerin bir araya gelip çoğalmasıyla oluşur. Bununla birlikte, bu yeni popülasyon, kendinden önceki popülasyonun özelliklerinin büyük bir kısmını taşımaktadır. Bu sayede, yeni nesiller boyunca iyi özellikler popülasyon içinde yayılırlar ve genetik işlemler aracılığıyla diğer iyi özelliklerle birleşirler.

Uygunluk değeri yüksek olan bireylerin bir araya getirilmesi ne kadar iyi sağlanırsa, arama uzayında o kadar iyi bir çalışma alanı elde edilir. Problemde aranan en iyi çözümün bulunabilmesi için, bireylerin gösterimi doğru yapılmalı, uygunluk fonksiyonu etkin bir şekilde oluşturulmalı ve doğru genetik işlemciler seçilmelidir. Bu kriterler yerine getirildiğinde, istenen çözüm kümesine en iyi yaklaşım sağlanır. Genetik algoritmalar, diğer optimizasyon yöntemlerine göre oldukça büyük arama uzayına sahip problemlerde başarı göstermektedir. Bir problemin en iyi çözümünü bulma garantisi vermez; ancak problemlere makul süre içinde, kabul edilebilir en iyi çözümü sunarlar.

Genetik algoritmalar, bilinen çözüm yöntemi olmayan problemler için çözüm aramak için kullanılır. Mutlak sonuçları ve kendine has çözüm teknikleri olan problemlerde genetik algoritma yöntemi tercih edilmemektedir. Fakat; arama uzayının çok büyük ve karmaşık olduğu, var olan bilgilerle sınırlı arama uzayında zor bir çözüme sahip olan, belirli bir matematiksel modelle ifade edilemeyen, geleneksel optimizasyon yöntemleriyle istenen sonucun alınamadığı problemler için etkili bir çözümdür.

4.2. Genetik Algoritma Gösterim Mekanizması

Genetik algoritma uygulamasında ilk olarak popülasyonu oluşturacak bireylerin nasıl gösterileceğine karar verilmesiyle başlanır. Bir birey, GA'nın örnek aldığı genetik biliminde olduğu gibi, bilgilerinin kodlandığı genlerin meydana getirdiği bir kromozom şeklinde tanımlanır [25].

Kodlama planı, GA için önemli bir aşamadır, zira bu plan kodlanan bilginin çerçevesini sınırlayabilir.

Probleme özgü bilgi, kromozomsal gösterimle temsil edilir ve genellikle problemdeki değişkenlerin belli bir düzende sıralanmasıyla sağlanır. Kromozomu oluşturmak için sıralanmış her bir değişkene gen adı verilir. Bu değişkenlerin gösteriminde en çok kullanılan mekanizma, ikilik sistemi kullanan bit dizisidir. Şekil 4.2'de genel olarak bir kromozomun ikilik sistem kullanılarak yapılmış kodlanma şekli verilmiştir. Şekilde görüldüğü gibi, kromozom farklı uzunluklara sahip genlerden oluşabilir, çünkü her bir genin türü ve içerdiği bilgi farklı olabilir.

Gen	1100100110	0010111010001	101011010011
	x_1 (10 bit)	x_2 (13 bit)	x_3 (12 bit)
Kromozom	1100100110001011101000110101101 0011		

Şekil 4.2. Kromozom kodlaması

4.2.1. Başlangıç popülasyonunun oluşturulması

Araştırma ve uygulama alanlarında karşılaşılan problemlerde, GA başlangıç yoğunluğu, genellikle rastgele sayı üreteci kullanılarak elde edilir. Ancak problem için başlangıçta bazı çözümler biliniyor ise, bu durumda başlangıç yoğunluğu bu çözümlerden oluşturulabilir. Böylece, optimal bir çözüm bulmada zaman açısından tasarruf sağlanır.

Popülasyon sayısının, şu eşitlik ile bulunması tavsiye edilmektedir [26]:

$$\text{Popülasyon sayısı} = 1.65 \times 2^{0.21 \times l} \quad (4.1)$$

Bu eşitlikte *l*, kromozom için kullanılan bit sayısını ifade etmektedir. Bir kromozom, problemi teşkil eden fonksiyondaki değişken sayısı kadar kodlanır. Kromozomu oluşturan ikili sayı dizisinin her bir kısmı bir değişkeni ifade etmektedir. Bu sayı dizisi içinde kullanılan ikili kodlamanın her bir sayısına gen adı verilir. Bir değişkenin, sahip olduğu sınırlar arasında alabileceği tüm değerler bu kodlamada yer alır. Böylece değişkenin hangi sınırlar arasında tarama yapacağı kodlama içinde belirlenmiş olur. Şekil 4.2'de verilmiş olan kodlama örneğinde, x_1, x_2 ve x_3 değişkenlerinden oluşan bir amaç fonksiyonunda, değişkenler 0 ve 1 sayıları kullanılarak kodlanmış, bu sayıların birleştirilmesiyle bir kromozom elde edilmiştir.

4.2.2. Uygunluk fonksiyonu ve amaç fonksiyonu

Başlangıç yoğunluğunun oluşturulmasının ardından, bireylerin evrimi başlar. Genetik algoritma için, bireylerin uygunluğuna göre ayrılıp farkedilmesine ihtiyaç vardır. Bir minimizasyon probleminde, en uygun bireyler amaç fonksiyonunu oluşturan değişkenlerin sayısal olarak en düşüklerinin birleştirilmesiyle elde edilir. Uygunluk fonksiyonu, uygunluk değerlendirmesine bağlı olarak amaç fonksiyonuna dönüştürmek için kullanılır.

Amaç fonksiyonu, problem alanı içerisindeki bireylerin performanslarının nasıl olduğunu görmek için kullanılır, ve her bir bireyin durumunu değerlendirmeyi sağlayan ana kaynaktır. Bu fonksiyon, genetik algoritma ve sistem arasındaki en önemli bağlantıdır. Amaç fonksiyonu girdi olarak kodlanmış bir kromozomu alır ve kromozomun performansına ölçü olacak bir objektif değer üretir. Bu işlemin tüm kromozomlar için yapılmasının ardından elde edilen değerler kullanılarak, uygun değerler uygunluk fonksiyonuyla

hesaplanıp belli bir düzende planlanır. Bu planlamayı sağlayan pencereleme ve lineer normalizasyon gibi bir çok yöntem vardır. Pencereleme yöntemi, en yaygın kullanılan normalizasyon işlemidir [26].

4.2.3. Seçim

Seçim fonksiyonu, evrim teorisinde yer alan doğal seleksiyon kavramından türetilmiştir. Genetik algoritmada, popülasyon içindeki gelecek nesli oluşturacak bireylerin seçilmesi, bireylerin kodlamasından sonraki ikinci işlemdir. Seçim işleminin amacı, popülasyondaki iyi bireyleri daha iyi özelliklere sahip çocuklara dönüştürmektir [27].

Seçim işlemiyle, bireylerin uygunluk değerlerine bakılarak, uygunluk değerleri yüksek olanlar düşük olanlara tercih edilir. Bu tercih, bir sonraki neslin oluşumunda elenenlerin yerini daha iyi bireylerin almasını sağlar. Bir popülasyon için eski bireylerden üstün özelliklere sahip olanlar seçilerek, çözüm kalitesi nesilden nesile aktarılır ve kötü bireylerin üreme şansı düşer. Bu süreç, istenen kriterler sağlanıncaya kadar bir döngü halinde devam ettirilir [28].

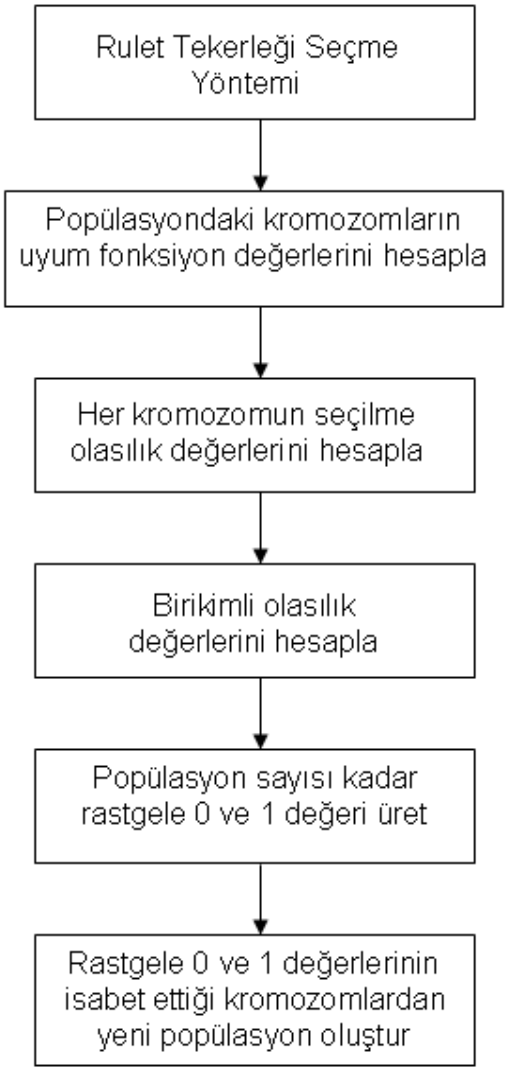
Seçim işlemi için kullanılan birden fazla teknik vardır. Bunlar; rulet tekerleği, turnuva seçimi, derecelendirmeli seçme, elitizmli seçme olarak sıralanmaktadır.

Rulet tekerleği

Rulet tekerleği yönteminde, kromozomların seçimi gittikçe artan olasılıkla yapılır. Her ne kadar seçme işlemi rastgele yapılıyor gibi görünse de, bireylerin seçimi uygunluk fonksiyon değeriyle kıyaslanır [29].

Rulet tekerleği seçim metodunun uygulanması için, ilk önce kromozomların toplam uygunluk değeri hesaplanır. Kromozomlar, toplam uygunluk değerine

bölünerek her bir kromozom için 0–1 arasında değişen seçim ihtimalleri bulunur. Daha sonra, kümülatif ihtimaller hesaplanır. Popülasyon sayısı kadar “rastgele” 0-1 arasında sayılar üretilir. Üretilen rasgele sayı, birinci kromozomun kümülatif seçim ihtimalinden küçük ise, birinci kromozom seçilir. Eğer değilse, ikinci kromozomun veya diğerlerinin kümülatif ihtimalleriyle karşılaştırılarak hangisinden küçükse o kromozom seçilir. Böylece rulet seçim metodu gerçekleştirilmiş olur. Rulet tekerleği metodunun akış diyagramı Şekil 4.3’de verilmiştir.



Şekil 4.3. Rulet Tekerleği Akış Şeması

Turnuva seçimi

Rulet tekerleği seçiminde, popülasyon üzerindeki bir bilgiye dayanarak bir olasılık dağılımından örnekleme yapılmaktadır. Fakat bazı özel durumlarda, örneğin popülasyonun çok büyük ya da çok dağıtık olması halinde, bu bilgiyi almak çok fazla zaman alır ve problemin çözümünü imkansızlaştırabilir.
Turnuva seçimi metodu, popülasyon hakkında global bir bilgiye ihtiyaç duymayan kullanışlı bir metoddur. Bu metodda yalnızca iki birey karşılaştırılarak bir sıralama bağıntısı oluşturulur. Dolayısıyla konsept olarak kurulumu ve uygulanması daha kolay ve hızlıdır. Turnuva seçiminde, sıralama yöntemi gibi mutlak uyuma değil, sadece bağıl uyuma bakıldığından uyum fonksiyonunun transpozesinin alınması durumunda sonuçlar değişecektir.

Turnuva seçiminde bir bireyin seçilmesi şu dört faktöre dayanır:

1) Popülasyon sırası. Ancak bu tüm popülasyonu sıralamadan tahmin edilir
2) Turnuva sayısı k. Turnuvaya Dahil edilen birey sayısı arttıkça turnuvanın sonucunda daha büyük uyuma sahip bireylerin çıkma ihtimali artar, dolayısıyla daha az uyuma sahip bireylerin çıkma ihtimali azalır.
3) En uygun bireyin turnuva sonrası seçilme olasılığı p. Genelde deterministik sistemlerde p=1 alınır, ancak stokastik yöntemlerde p<1 olarak kullanılır. İkinci durumda seçim baskısı doğal olarak daha azdır.
4) Turnuva metodu deterministik ve yer değiştirmeli olarak uygulanıyorsa, en az uyumlu bireyin seçilme ihtimali yoktur. Fakat yer değiştirmeli bir seçim yapılıyorsa, eğer şanslı ise en az uyumlu bireyin seçilebilme ihtimali her zaman vardır.

Derecelendirmeli seçme

Derecelendirmeli seçme yöntemi, amaç fonksiyonunun bir noktada kesişmesini hızlı ve pratik bir şekilde yapmak için kullanılan alternatif bir seçme yöntemidir. Amaç fonksiyonunun değerlerine göre popülasyondaki bireylerin sıralanması sonucu kromozomların seçilme olasılıkları, tüm bireyler arasındaki uygunluk durumundan daha ziyade, her bireyin sıralamadaki yerine bağlı olan seçimdir [27].

Elitizmli seçme

Elitizmli seçme yönteminde, bir sonraki nesil için en iyi kromozoma hayatta kalma garantisi verilir. Böylece, en iyi kromozomun popülasyon için uygulanan çoğalma, çaprazlama ve mutasyon operatörlerinden dolayı kaybedilmemesi sağlanır. Bir çok araştırma sonucunda, elitizmin genetik algoritmanın performansını önemli ölçüde geliştirdiği gösterilmiştir [27].

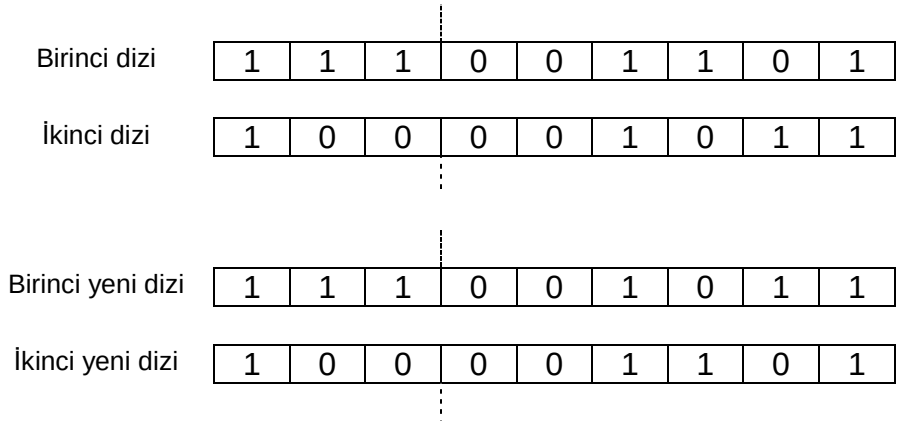
4.2.4. Çaprazlama

Çaprazlama işlemi, genetik biliminde yer alan çaprazlama tekniğinden uyarlanmıştır. Genetik çaprazlamayla ortaya çıkan melez bireylerin üretilmesinde kullanılan genetik algoritmaya eşdeğer bir fonksiyon yapısı vardır. Eşleştirme havuzunda yer alan bireylerden birer çift seçilir ve çaprazlama yoluyla bu iki bireyden yeni iki birey meydana getirilir. Eski bireyler (ebeveyn) ve çaprazlama işleminin gerçekleştirilmesinden sonra ortaya çıkan yeni iki birey (çocuk), mevcut nesilde ya tutulurlar ya da birbirleri ile yer değiştirirler. İkinci durumda, kötü bireyler atılır ve yoğunluk büyüklüğü sabit olarak korunur. Böylece eldeki nesilden farklı nesiller elde edilmiş olur. Oluşan yeni bireyler, ebeveyn bireylerin bazı özelliklerini almış, ve bir anlamda ikisinin kopyası olmuştur.

Çaprazlama işlemi için birden fazla teknik mevcuttur [30].

Tek noktalı çaprazlama

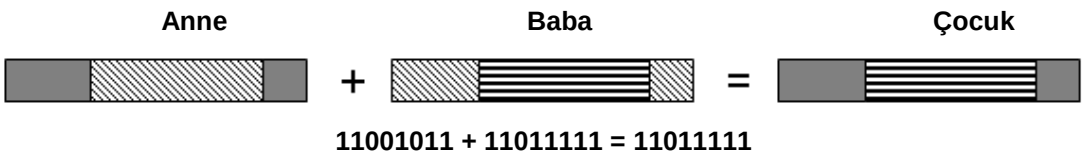
Tek noktalı çaprazlama, en basit çaprazlama şeklidir. Eski bireylerin kodlama yapısı üzerinde rastgele bir nokta seçilir ve buradan kesilir. Bireylerin kesilen uzantıları yeni iki yapı üretmek için birbirleri arasında yer değiştirir. Bu şekilde yapılan çaprazlama işlemi Şekil 4.4.'de gösterilmektedir.



Şekil 4.4. Tek noktalı çaprazlama

Çok noktalı çaprazlama

Çok noktalı çaprazlamanın tek noktalı çaprazlama işleminden farkı, kesim işleminin kodlama üzerinde birden farklı yerden yapılmasıdır. İki noktalı kesim yapılacak ise, kromozomun ilk ve son kesimleri olduğu gibi kalır, orta kesim yer değiştirir. İkiden fazla kesim yapılacağı durumda, ilk kesim aynen kalırken ikinci kesimler kromozomlar arasında yer değiştirir. Yine devamında üçüncü kesimler aynı kalırken, dördüncü kesimler yer değiştirir. Bu şekilde, devamındaki kesimlerde yine aynı sırayla kesimler birer atlayarak yer değiştirir (Şekil 4.5).



Şekil 4.5. Çift Noktalı çaprazlama

Düzenli çaprazlama

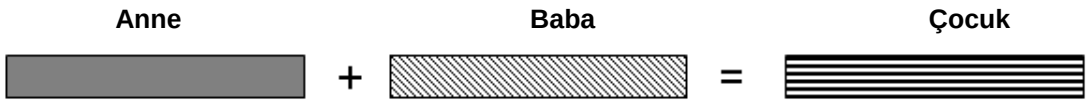
Düzenli çaprazlama tekniğinde, ebeveyn bireylerden alınan bitler rastgele seçilerek bir sonraki nesile kopyalanır. Bu işlemde, kromozomun bölüneceği nokta olmadan, kromozomun kendi uzunluğu kadar rastgele oluşturulmuş bir şablonla kromozomlar arası yapılan genlerin çocuk bireylere taşınması gerçekleştirilir. Bu işlem, rastgele oluşturulan şablonun genlerine bakarak ebeveyn bireylerin genlerini çocuklara aktarmak anlamına gelir. Şablon kromozom üzerinde, eğer ikili kodlamaki bit 1 ise anne ebeveynden kromozom alınarak, eğer 0 ise baba ebeveynden gen alınarak çaprazlama işlemi gerçekleştirilir. İkinci çocuk birey için ise, bu sefer 1 ise babadan, 0 ise anneden gen alınır. Bu şekilde hem anne kromozomun hem de baba kromozomun genleri çocuk kromozomlara verilmiş olur (Şekil 4.6).

11001011 + 11011111 = 11011111

Şekil 4.6. Düzenli çaprazlama

Aritmetik çaprazlama

Bu çaprazlama tekniğinde, ebeveynlerin genleri çeşitli aritmetik işlemlere tabi tutulur ve yeni nesil bireylerin oluşması sağlanır (Şekil 4.7).



11001011 + 11011111 = 11001001 (VE)

Şekil 4.7. Aritmetik çaprazlama

4.2.5. Mutasyon

Mutasyon işleminde, çarprazlama operatöründe olduğu gibi biyoloji biliminde yer alan genetik mutasyon olayı simüle edilmiştir. Mutasyon genetik algoritmanın başarısında önemli işlemlerden biridir. Bu işlemde, yeni bir genetik yapı oluşturmak için iki kromozom alınır ve kromozomların herhangi bir geni karşılıklı olarak yer değiştirir. Mutasyon işlemiyle, kromozomların sürekli aynı geometrik yerde dolaşmaları ve geçmiş görevlerini tekrarlamaları ihtimaline karşı popülasyon sağlama alınmış olur.

Mutasyon sayesinde, genetik bilginin kodunun değiştirilmesiyle kromozomların hep aynı gen yapısında olması engellenir. Çaprazlamayla elde edilemeyen, genetik bilgisi iyi durumdaki kromozomları mutasyonla elde etmek mümkün olmaktadır. Ancak mutasyon işleminin, bir kromozomun çok yüksek genetik bilgi uygunluğuna sahip olduğu durumda bu yapıyı bozma ihtimali de bulunmaktadır [31].

Genetik algoritmada, mutasyon işlemi düşük olasılıklarla uygulanır. Ancak mutasyon olasılığının uygun seçimi önemlidir. Bu olasılık, mutasyon işleminin lokal bir noktaya takılmasını engelleyecek derecede yüksek, ancak çaprazlama ve çoğullama işlemlerinin getirdiği en iyi noktaya gidişi engellemeyecek ölçüde düşük seçilmelidir [32].

Mutasyon işleminin birden fazla tekniği bulunmaktadır.

<u>Ters çevirme mutasyonu</u>

Ters çevirme mutasyonunda, şekil 4.8’de görüldüğü gibi kromozom içinde rastgele iki gen seçilir ve yerleri değiştirilir.

Çocuk	3	8	**4**	**7**	**0**	**9**	5	2	1	6	8
Mutasyonlu Çocuk	3	8	**4**	**7**	**0**	**9**	5	2	1	6	8

Şekil 4.8. Ters Çevirme Mutasyonu

Ekleme mutasyonu

Bu teknikte, Şekil 4.9'da görüldüğü gibi rastgele bir parça seçilir ve seçilen parça rastgele bir yere yerleştirilir (Şekil 4.9).

Çocuk	3	8	4	7	0	9	5	**2**	1	6	8
Mutasyonlu Çocuk	3	8	2	4	7	0	9	5	1	6	8

Şekil 4.9. Ekleme Mutasyonu

Yer değiştirme mutasyonu

Yer değiştirme mutasyonunda, Şekil 4.10'da görüldüğü gibi rastgele bir alt dizi seçilir ve rastgele başka bir alt dizi ile yer değiştirilir.

Çocuk	3	8	**4**	**7**	**0**	**9**	5	2	1	6	8
Mutasyonlu Çocuk	3	8	5	2	1	**4**	**7**	**0**	**9**	6	8

Şekil 4.10. Yer Değiştirme Mutasyonu

Karşılıklı değişim mutasyonu

Bu teknikte, Şekil 4.11'de görüldüğü gibi rastgele iki gen seçilir ve birbirleri arasında yer değiştirilir.

Çocuk	3	8	**4**	7	0	9	5	**2**	1	6	8
Mutasyonlu Çocuk	3	8	**2**	7	0	9	5	**4**	1	6	8

Şekil 4.11. Karşılıklı Değişim Mutasyonu

4.3. Genetik algoritmanın uygulama alanları

Goldberg tarafından genetik algoritmanın pratik uygulamalarının gerçekleştirilebileceğinin gösterilmesinin ardından [23], bu teknik çok çeşitli alanlarda uygulanmaya başlanmıştır. Genetik algoritmanın günümüzdeki uygulamaları üç ana grupta toplanır. Bunlardan ilki deneysel uygulamalardır. Mevcut diğer optimizasyon algoritmalarına karşılık olarak, genetik algoritmanın üstünlüğünü ispat etmek amacıyla belli problem çözümlerini bulmak için deneysel uygulamalarda kullanılmaya başlanmıştır. İkinci olarak, genetik algoritmalar pratik uygulamalarda yer almaktadır. Endüstriyel ve diğer gerçek problemlerin çözümünde, çok farklı alanlarda genetik algoritma kullanılmaktadır. Üçüncü ve son grupta ise, sınıflandırıcı sistem uygulamaları yer almaktadır. Bu uygulamalarda genetik algoritma, bilgi çıkarılması amacıyla kullanılmaktadır.

Deneysel uygulamalar grubuna örnek olarak gezgin satıcı problemi, kör Knapsack problemi, çift kollu ve k kollu yol kesme problemleri ve grafik bölme problemleri verilebilir [33]. Pratik uygulamalar grubu için, sayısal optimizasyon problemleri, tablo problemleri, yerleşim problemleri ve görüntü işleme uygulamaları örnek verilebilir [34]. Üçüncü grup olan bilgi çıkarılması alanında ise, bir uzman sistemin bilgi tabanını oluşturan kuralların elde edilmesi örnek gösterilebilir [35].

5. DAİRESEL MİKROŞERİT ANTEN DİZİSİNİN GENETİK ALGORİTMA YOLUYLA OPTİMİZASYONU

5.1. Giriş

Bu çalışmada, dairesel mikroşerit yapısına sahip antenlerden düzlemsel olarak bir dizi oluşturulmuş ve bu dizinin ışıma örüntüsü genetik algoritma tekniği kullanılarak optimize edilmiştir. Oluşturulan anten dizisinin istenilen doğrultuda maksimum ışımayı sağlaması için gerekli parametrelerin bulunması işleminin klasik formülizasyon yöntemleriyle gerçekleştirilmesi zordur. Diziyi oluşturan her bir antenin hangi faz değerine sahip olacağı, deneme yanılma yoluyla elde edilebilir. Optimizasyon işlemi için üstünlüğü çeşitli çalışmalarla kanıtlanmış olan genetik algoritma işlemi, pratik ve bilimsel çalışmalarda tercih edilmeyen deneme yanılma metodunun yerine kullanılmıştır.

Çalışmada C# programlama dili kullanılmıştır. Bu programlama dilinin günümüzde kullanımı giderek artmaktadır ve C programlama dilinin türevidir. C# programlama dili; kodlama esnasında kullanım kolaylığı sağlaması, görsel olarak daha iyi arayüzler elde edilebilmesi ve çıktısı grafiksel olan sonuçların elde edilmesinde başarı sağlaması nedeniyle bu çalışmada tercih edilmiştir.

5.2. Problemin tanımlanması

Bu çalışmada, 5 satır ve beş sütun olmak üzere toplam 25 dairesel mikroşerit antenden oluşan bir anten dizisi modellenmiştir. Bu antenlerin modellenmesi için, daha önceki bölümlerde Eş. 2.12b ve Eş. 3.24 ile verilen elektrik alan ve dizi faktörü eşitlikleri kullanılmıştır. Anten dizisinin elemanları arasındaki uzaklık değişebilir olmakla birlikte, problemde antenler arasındaki uzaklığın eşit olduğu varsayılmıştır.

5.2.1. Tek bir antenin ışıma örüntüsünün elde edilmesi

Tek bir dairesel mikroşerit anten elemanın elektrik alan formulü, ikinci bölümde verilen Eş. 2.12b eşitliği yoluyla elde edilir:

$$E_\theta = j\frac{k_0 a_e V_0 e^{-jk_0 r}}{2r}\left[J'_{02}\right] \tag{5.1}$$

Bu ifadede, k_0, V_0 ve r değerleri sabit olup, hesaplamalarda normalizasyon işleminin gerçekleştirilmesi nedeniyle antenin ışıma örüntüsünü etkilememektedir. a_e yarıçap değeri için ise, hazırlanan programda istenilen değer girilebilmektedir.

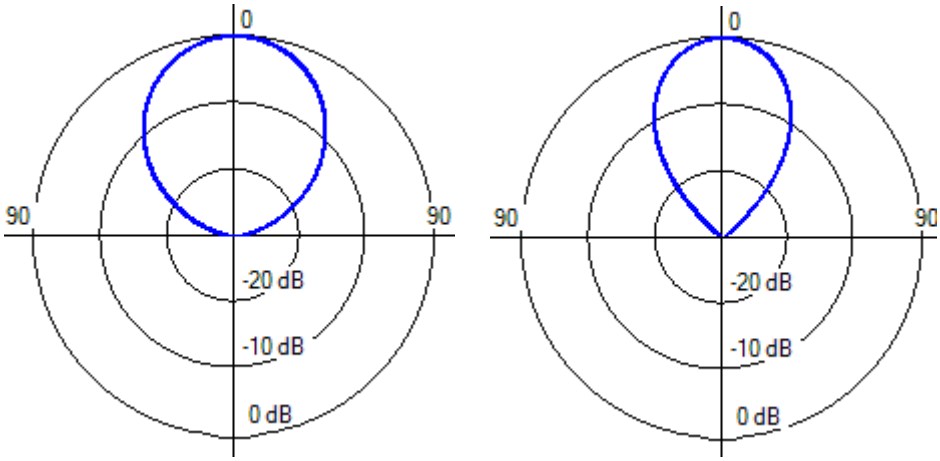
J'_{02} ifadesi ise Bessel fonksiyonunu temsil etmekte olup, açılımı aşağıdaki gibidir.

$$J'_{02} = J_0(k_0 a_e \sin\theta) - J_2(k_0 a_e \sin\theta) \tag{5.2}$$

J_0 ve J_2 değerlerinin elde edilebilmesi için programda Bessel fonksiyonları yazılımsal dile çevrilmiştir. Bessel fonksiyon kodlarından elde edilen değerler çeşitli kaynaklarda verilen Bessel tablolarıyla karşılaştırılmış doğruluğu teyid edilmiştir [2].

Verilen eşitliklerin yazılım diline çevrilmesinin ardından elde edilen değerler programda grafik komutları yoluyla kullanılarak tek bir elemanın ışıma örüntüsünün grafiksel olarak elde edilmesi sağlanmıştır.

İdeal bir dairesel mikroşerit antende, ışıma örüntüsünü etkileyen faktör antenin yarıçapıdır. Bu durum, bu tür antenin modellemesini kolaylaştırmakta ve tercih edilebilirliğini artırmaktadır. 10 GHz çalışma frekansında farklı a_e değerleri için elde edilen ışıma örüntüleri Şekil 5. gösterilmektedir.

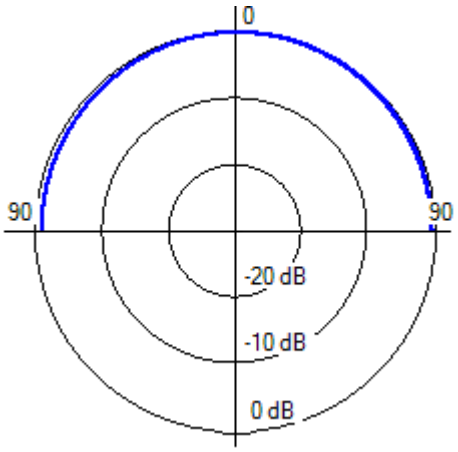


(a) a_e = 0.75 cm (b) a_e = 1 cm

Şekil 5.1. Farklı yarıçap değerleri için elde edilen ışıma örüntüleri

Şekil 5.1 (a)'da 0.75 cm yarıçapına sahip bir anten huzmesi görülmektedir. Şekil 5.1 (b)'de antenin yarıçapı artırılmış ve 1 cm yarıçapına sahip bir antenin ışıma örüntüsü sunulmuştur. Örneklerde görüleceği üzere, anten yarıçapı arttıkça ana ışıma huzmesinin daraldığı görülmektedir. Anten huzmesinin daralması antenin daha iyi yönlendirilmesini sağlayabilir. Ancak antenin dizi oluşturularak bir çok yönde tarama yapması istendiğinde, dar bir ışıma huzmesine sahip anten diğer yönlerde ışıma yapamayacaktır. Bu nedenle anten dizisini oluşturan anten elementinin ışıma örüntüsünün izotropik yapıya sahip olması daha uygundur. Antenin yarıçapı azaltıldıkça, izotropik ışıma yapısına daha çok yaklaşılır. Ancak kullanılacak anten yarıçapı hesabı yapılırken, antenin üretim aşamasında zorluk yaratmayacak küçüklükte olmamasına dikkat edilmelidir.

Çalışmada anten ışıma örüntüleri elde edilirken tüm değerler 0 dB ile -30 dB arasında olacak şekilde hesap edilmiştir. Göreceli güç ya da dB down olarak adlandırılan bu çizim şeklinde maksimum değer 0 dB'den başlar ve 10 dB aralıklarla -30 dB değerine kadar logaritmik olarak sunulur. Grafiklerde -30 dB'den düşük değerlerin gösterimi yapılmamaktadır. Göreceli güç gösterimi, antenin ışıma örüntüsünün görsel olarak daha efektif sunulmasına yardımcı olmaktadır.

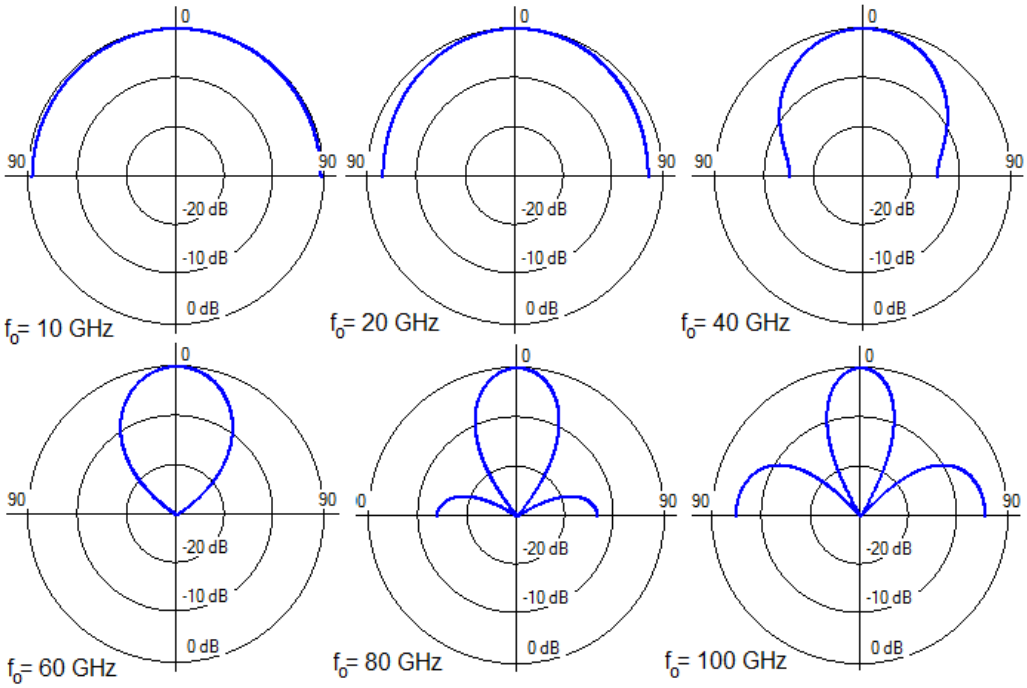


Şekil 5.2. a_e = 0.15 cm durumunda dairesel antenin ışıma örüntüsü

Tez çalışması için yapılan modellemede, a_e yarıçap değeri 0.15 cm olarak alınmış, antenin çalışma frekansı 10 GHz olarak belirlenmiştir. Bu durumda elde edilen ışıma örüntüsü Şekil 5.2'deki gibidir. Bu grafikler elde edilirken Φ = 0 düzleminin kullanıldığına dikkat edilmelidir. Şekil 5.2'de görüldüğü gibi, kullanılan yarıçap değerinde, her yöne yaklaşık olarak eşit şiddette yayılım yapan bir anten elemanı elde edilmektedir. Böylece 0.15 cm yarıçapına sahip özdeş mikroşerit dairesel antenler kullanılarak oluşturulan anten dizisinde, anten yönlülüğü değiştirildiğinde antenin ışıma örüntüsü anten dizisinin ışıma örüntüsüne fazla etki etmeyecektir. Bu nedenle, anten dizisinin ışıma örüntüsü elde edilirken, dizi faktörü önemli bir faktör olacaktır.

Şekil 5.3'de farklı f_o çalışma frekanslarında a_e = 0,15 cm yarıçapa sahip mikroşerit dairesel antenin ışıma örüntüleri görülmektedir. Örüntüler karşılaştırıldığında, frekans değeri arttıkça ışıma örüntüsünün izotropik yapıya sahip olmaktan çıkıp ana hüzmesinin daraldığı görülmektedir. Bunun yanısıra ana hüzmenin dışında yan loblar oluşmakta ve frekans değeri arttıkça yan lobların büyüklüğü artmaktadır. Anten dizisi oluşturulurken tek bir yöne ışıma yapılması istendiği durumlarda büyük yan loblara sahip bir ışıma örüntüsü olumsuz sonuçlar verir. Yüksek frekanslarda geniş bir ana hüzmeye sahip bir örüntü elde etmek için anten yarıçapının azaltılması gerekir. Aynı

şekilde, düşük frekanslarda da anten yarıçapının artırılması olumlu sonuçlar verir.



Şekil 5.3. Farklı çalışma frekanslarına sahip dairesel antenin örüntüleri (a_e = 0,15 cm)

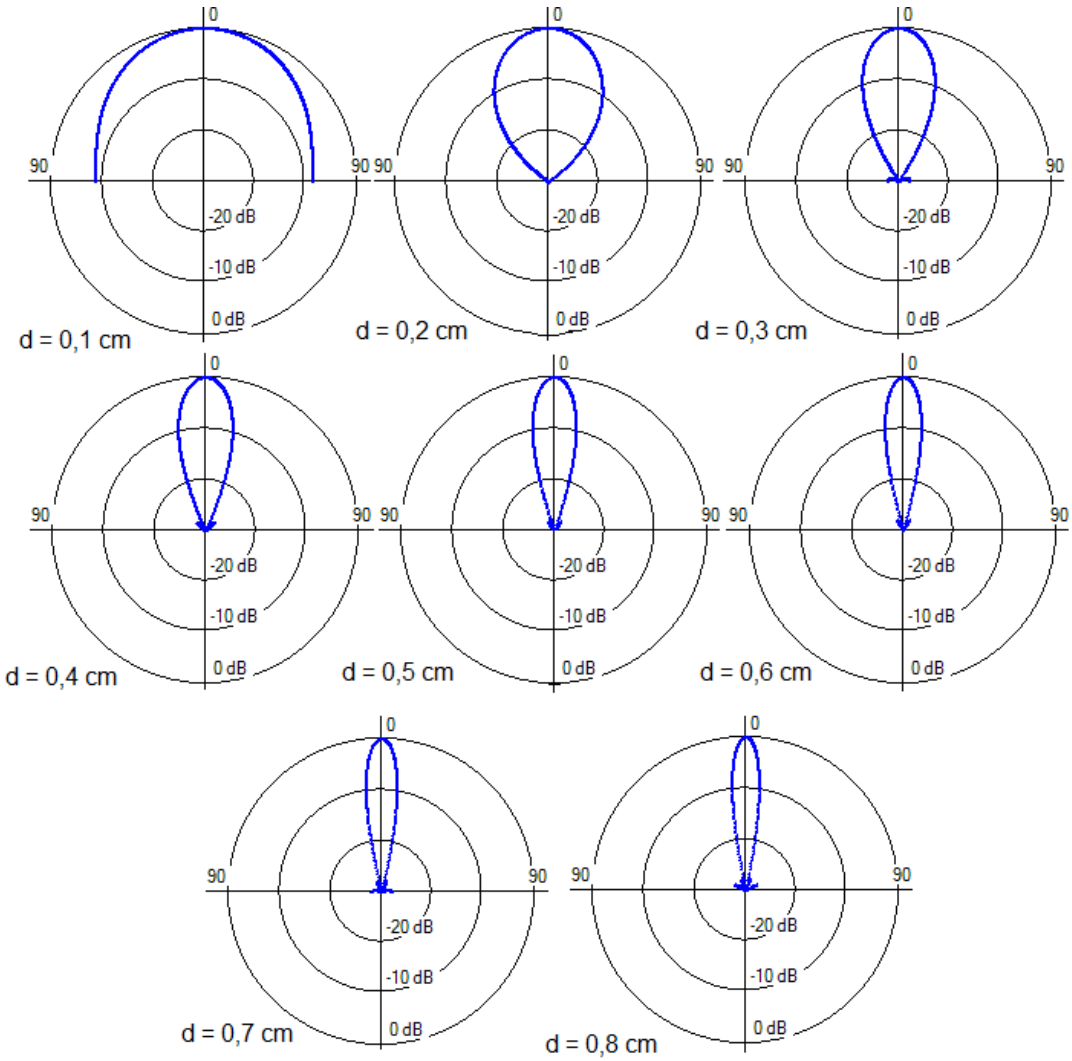
5.2.2. Dizi faktörünün elde edilmesi

Düzlemsel anten dizisinde, dizi faktörünü elde etmek için üçüncü bölümde verilen Eş. 3.24 eşitliği kullanılmıştır.

$$AF(\theta,\phi) = \sum_{i=1}^{N} \alpha_i e^{j\beta_i} e^{j2\pi dx_i \sin\theta\cos\phi} e^{j2\pi dy_i \sin\theta\sin\phi} \qquad (5.3)$$

Bu eşitlikte α_i değeri sabittir. Dizinin ışıma örüntüsünü etkileyecek önemli faktörler dx_i, dy_i ve β değerleridir. Tek bir anten elemanının ışıma örüntüsünün elde edilmesinde olduğu gibi, dizi faktörünü elde ederken de $\Phi = 0$ alınmalıdır. $\Phi = 0$ düzleminde $\sin\Phi$ çarpanı sıfıra eşit olacağından anten elemanlarının x eksenine uzaklığını ifade eden dy_i çarpanı etkisiz hale

gelmektedir. Tüm elemanların fazlarının sıfıra eşit olduğu durumda farklı dx_i değerleri için elde edilen dizi faktörleri Şekil 5.4'de verilmektedir.

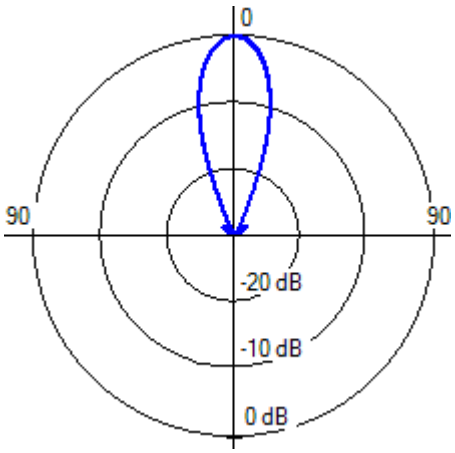


Şekil 5.4. Farklı anten arası mesafe değerleri için dizi faktörü örüntüsü

Bu dizilerde, bütün anten elemanları arasındaki uzaklıklar eşit olarak ele alınmıştır. Grafikler sadece AF bileşeni kullanılarak elde edildiğinden anten elemanı grafiğe etki etmemektedir. Tüm elemanların faz değerleri 0 olarak alındığından dizi faktörü örüntüsü 0˚ doğrultusunda yönlülüğe sahiptir. Farklı uzaklıklar için örüntüler karşılaştırıldığında antenler arası uzaklık arttıkça ışıma huzmesinin daraldığı görülmektedir. Dar bir ana ışıma huzmesi yönlülüğün sağlanmasında etkili sonuçlar verir. Ancak antenler arası

uzaklıklar artırıldıkça, yan ışıma huzmelerinin oluştuğu ve büyüdüğü görülmektedir. Bu da ışıma örüntüsünün tek bir yöne doğru yapılması istendiğinde olumsuz sonuçlar doğurur. Bu nedenle tasarlanacak anten dizisinde antenler arası uzaklığın en uygun şekilde seçilmesi önemliir.

Çalışmada kullanılan anten dizisi modelinde, dx_i = 0.4 cm olarak alınmıştır. Bu durumda elde edilen dizi faktörü örüntüsü Şekil 5.5'deki gibidir.

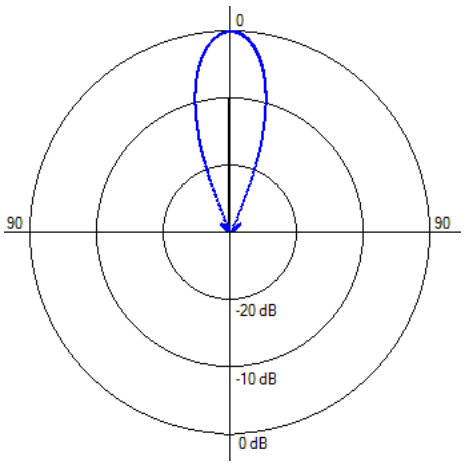


Şekil 5.5. d=0.4 cm durumunda dizi faktörü örüntüsü

Anten dizisinde kullanılan bütün elemanlar özdeş olarak kabul edildiğinden, elde edilen dizi faktörü tek bir antenin ışıma örüntüsü ile çarpılarak dizinin toplam ışıma örüntüsü elde edilir. Bu durum, her doğrultuda antenin elektrik alan değeri ile dizi çarpanı değerinin çarpıldığı şu eşitlikle tanımlanmıştır:

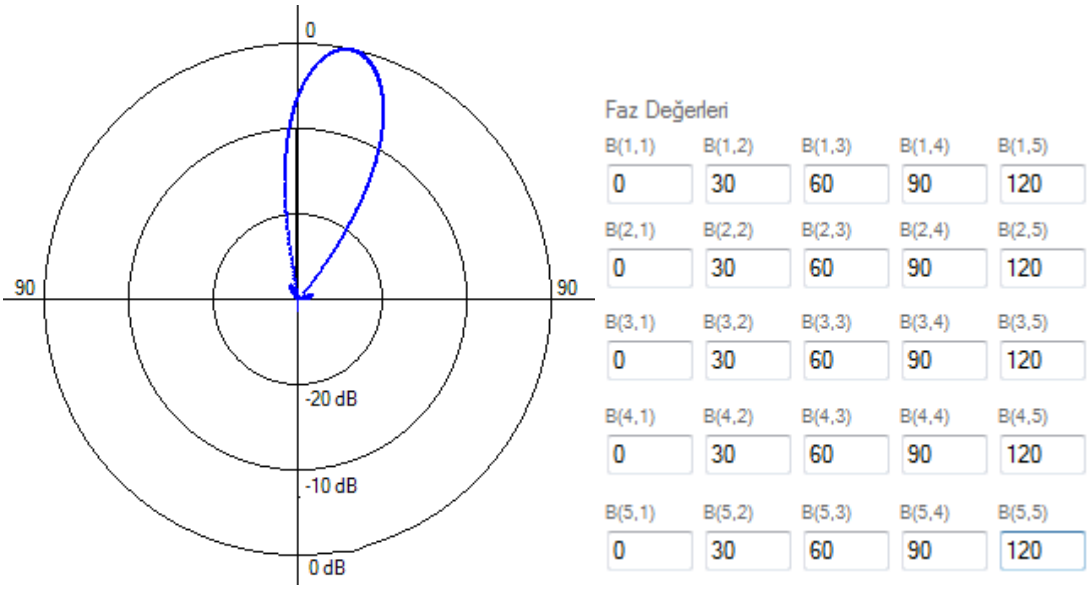
E(toplam) = [E(tek bir eleman) x [AF(dizi çarpanı)] (5.4)

Tek bir antenin ışıma örüntüsü Şekil 5.2'de verildiği gibi her yöne yaklaşık olarak eşit ışıma gerçekleştiren bir örüntü yapısına sahiptir. İzotropik yapıya sahip anten elemanının elektrik alan değeri dizi çarpanı ile çarpıldığında anten dizisinin toplam elektrik alanı elde edilir. Seçilen anten elemanı her yöne eşit ışıma yaptığından, anten dizisinin toplam ışıma örüntüsü, dizi faktörünün örüntüsü ile Şekil 5.6'de görüleceği gibi benzer olacaktır.

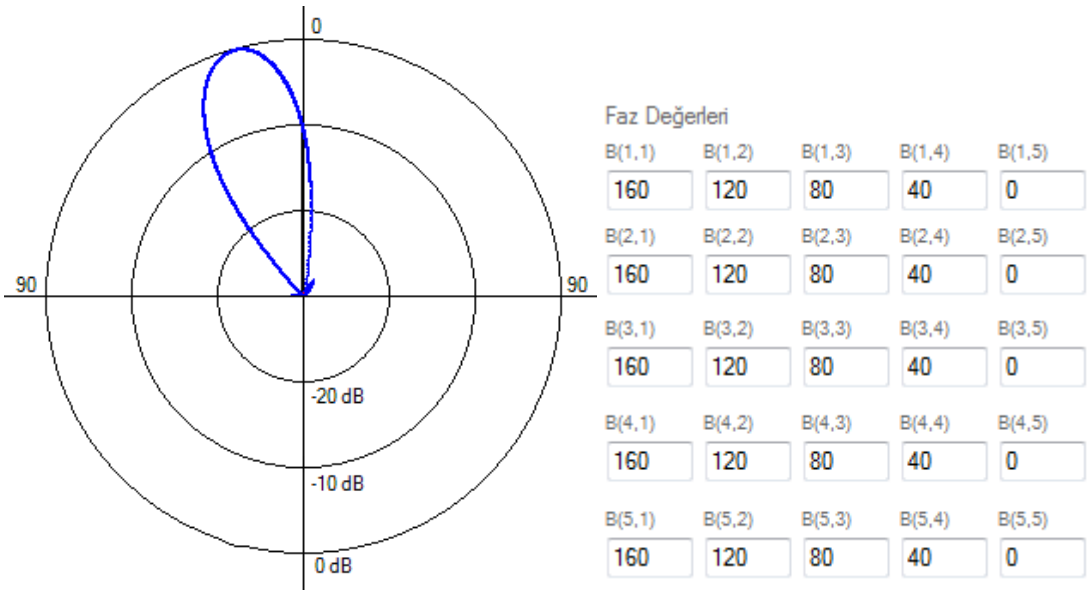


Şekil 5.6. Modellenen anten dizisinin toplam ışıma örüntüsü (a_e =0.15 cm, dx_i = 0.4 cm, Φ = 0)

Şekil 5.7 ve Şekil 5.8'de, farklı faz değerlerine sahip anten elemanlarından oluşan anten dizilerinin ışıma örüntüleri verilmektedir. Şekillerin sağında yer alan değerler, birbirine eşit uzaklıktaki mesafelerle yerleştirilmiş her bir anten elemanının faz değerini göstermektedir. Sol kısımda yer alan ışıma örüntülerinin elde edilmesi için faz değerleri kutucuklara manuel olarak girilmiştir.



Şekil 5.7. Farklı faz açılarına sahip anten dizisinin ışıma örüntüsü (a_e = 0.4 cm, Φ = 0)



Şekil 5.8. Farklı faz açılarına sahip anten dizisinin ışıma örüntüsü (a_e = 0.4 cm, Φ = 0)

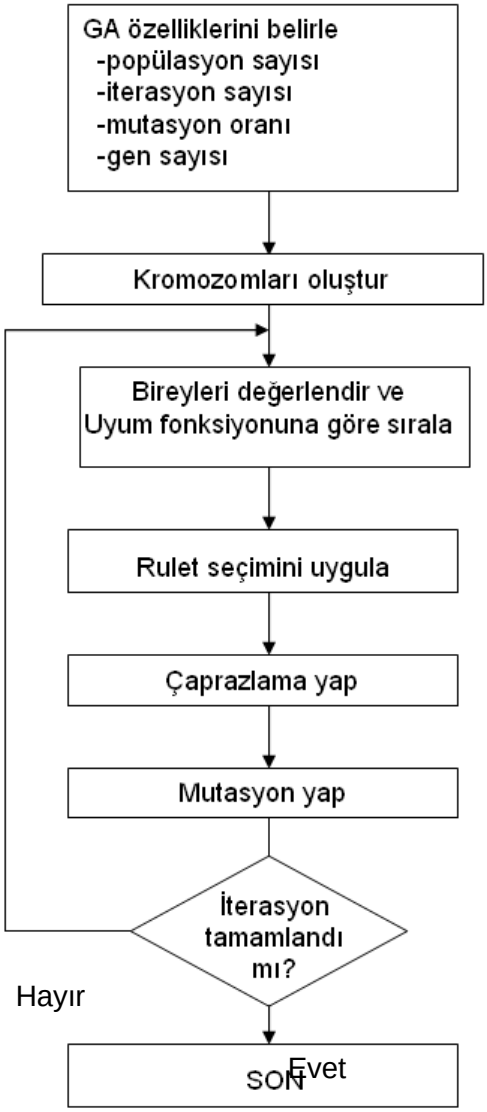
Her iki şekilde de 5x5 büyüklüğünde bir anten dizisi kullanılmaktadır. Şekil 5.7'de, y eksenine eşit uzaklıklardaki anten elemanları için aynı faz değerleri her bir sıra için artacak şekilde atanmıştır. Değerler rastgele seçilmiş olup, anten elemanlarının fazları değiştirilerek maksimum ışıma doğrultusu değiştirilebildiği gösterilmek istenmiştir. Şekil 5.8'de ise değerler yine rastgele seçilmiştir ve bir önceki anten yapılanmasından farklı olarak faz değerleri azalacak şekilde atanmıştır. Faz değerleri artan şekilde seçildiğinde, ana huzmenin sağa doğru yönlendiği, azalan şekilde seçildiğinde sola doğru yönlendiği görülebilmektedir. Farklı faz değerleri seçilerek ışıma örüntüsüne istenilen doğrultularda yön vermek mümkündür. Bu seçim işleminin deneme yanılma yoluyla yapılması mümkün olmadığından, istenilen örüntülerin elde edilebilmesi için optimizasyon işleminin gerçekleştirilmesi gerekmektedir.

Tez çalışması için oluşturulan anten dizisinde, istenilen doğrultuda maksimum ışımanın sağlanması için gereken parametrelerin belirlenmesi, genetik algoritma optimizasyonu yoluyla gerçekleştirilmiştir.

5.3. Problemin genetik algoritma yoluyla çözümü

Modellenen anten dizisinin ışıma örüntüsü elde edildikten sonra elde edilen veriler, problemin çözümünde genetik algoritma için popülasyon oluşturmak için kullanılacaktır.

Modellenen anten dizisinde istenen ışıma örüntüsünü verecek değerlerin elde edilmesini sağlayan genetik algoritmanın akış şeması Şekil 5.9'da verilmiştir.



Şekil 5.9 Programda uygulanan GA akış diyagramı

5.3.1. Uyum fonksiyonu

Uyum fonksiyonunu oluşturmak, problemin en iyi çözümünü elde etmek için gereken önemli adımlardan birisidir. Anten dizisinin toplam elektrik alanı, küresel koordinatlarda θ ve Φ açılarına bağlı olarak değilmektedir. Grafiksel gösterimde, Φ = 0 alınmış, θ değeri 0 ile 360 derece arasında değiştirilerek ışıma örüntüsü elde edilmiştir.

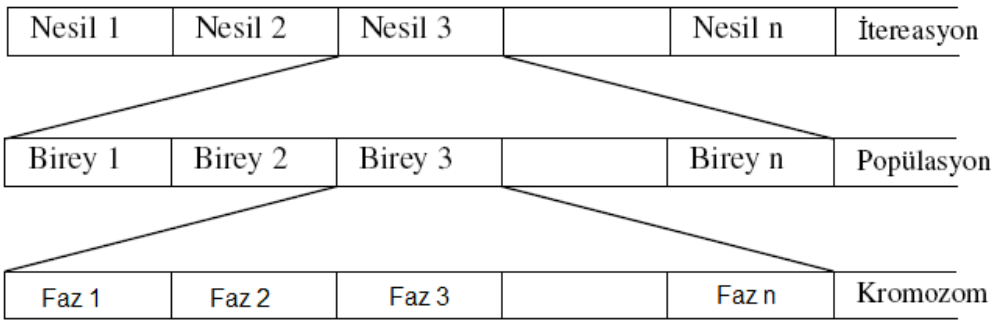
Problemde, antenin maksimum ışıma yapması istenen doğrultu açısal olarak tanımlanır. Genetik algoritma işlemi, verilen açıda maksimum ışımayı sağlayacak faz değerlerini verecektir. Bu nedenle uyum fonksiyonundan, θ için 0'dan 360 dereceye kadar olan değerler arasından, verilen açı için maksimum değere sahip olan örüntünün önem derecesini yüksek olarak vermesi istenir. Bu nedenle uyum fonksiyonu aşağıdaki gibi belirlenmiştir.

Uyum fonksiyonu = E(verilen açı) / Σ(θ = 0 - 360) E(θ,Φ) (5.5)

Bu fonksiyondan dönen değerin büyüklüğü, verilen açıdaki ışımanın tüm yönlere yapılan toplam ışımaya göre ne kadar büyük olduğunu göstermektedir. Farklı faz açıları için bu değer değişmektedir. Problemde istenen ise, bu değerin maksimum olduğu faz değerlerini elde etmektir.

5.3.2. Kromozomların kodlanması

Problemde, anten dizisinin elemanlarının yarıçapları ve elemanlar arasındaki uzaklıklar sabit tutulmuştur. Bu değerlerin optimizasyonu yapılmayacağından kromozom kodlamasında yer almamaktadırlar. Şekil 5.10'da problem için tanımlanan kromozom şekli yer almaktadır.



Şekil 5.10. Kromozom Kodlamasının Gösterimi

Optimize edilecek olan değerler, dizi elemanlarının faz değerleri olduğu için, her bir kromozom anten dizisindeki tüm elemanların faz değerlerinin ardarda eklenmesiyle oluşturulmuştur. Problemde 5x5 büyüklüğünde bir anten dizisi modellendiğinden, her bir kromozom yirmi beş adet faz değerinden oluşmaktadır.

5.3.3. Başlangıç popülasyonunun oluşturulması

Kromozomların kodlanmasının ardından, GA'nın aşamalarından biri olan başlangıç popülasyonunun oluşturulması işlemi gerçekleştirilir. Her bir kromozom popülasyon için bir birey teşkil etmektedir. Popülasyon için kromozomlar oluşturulurken genleri teşkil eden faz değerleri rastgele sayılara atanarak elde edilmektedir. Dolayısıyla kromozomlar da rastgele elde edilmiştir. Kromozomların oluşturulmasında yapılan tek kısıt, faz değerlerinin 0-360 derece arasında olmasını sağlamaktadır. Ancak farklı problemlerde belirli şartlara bağlı olarak farklı kısıtlar da konulabilir.

Popülasyonun birey sayısının artırılması, daha geniş örnekleme sağlayarak en iyi çözüme yakınlaştırsa da, birey sayısı arttıkça programın hesaplama süresi artmaktadır. Bu iki durum göz önüne alınarak, optimum popülasyon büyüklüğü seçilmelidir. Bu çalışmada, ilk popülasyondaki bireyler 100 adet olarak seçilmiştir.

5.3.4. Seçim işlemi

Çalışmada, başlangıç popülasyonunun oluşturulmasının ardından bir sonraki nesli elde etmek için, seçim metodu olarak rulet tekerleği kullanılmıştır. Bu yöntem sayesinde, uyum fonksiyonundan daha yüksek değerle dönen bireylere daha fazla seçilme olasılığı tanınır.

Seçim işleminin gerçekleştirilmesinin ardından, 50 adet anne birey ve 50 adet baba birey seçilmiş olur.

5.3.5. Çaprazlama

Seçim işlemiyle 50 adet anne ve 50 adet baba birey oluşturulduktan sonra, daha iyi niteliklere sahip bireyler elde etmek için bu iki grup arasında çaprazlama işlemi gerçekleştirilir. Çaprazlama işleminin uygulanma olasılığı arttıkça, elde edilen bireylerin çeşitlilik düzeyi de artar.

Bu çalışmada, çaprazlama yöntemlerinden biri olan tek noktalı çaprazlama işlemi yapılmıştır. Çaprazlama oranı %80 olarak belirlenmiştir. Bir önceki adım olan seçim işleminde elde edilen anne ve baba bireylerden genler rastgele alınarak bir sonraki nesili teşkil eden çocuk bireyler oluşturulmuştur. 50 anne ve 50 baba bireyden toplamda 100 çocuk birey elde edilmiştir.

5.3.6. Mutasyon

Çaprazlama işlemi yeni oluşturulan popülasyondaki çeşitliliği sağlasa da, bireyler genlerin değiştirilmesi yoluyla elde edildiği için birbirinin benzeri ya da aynısı bireyler elde edilme ihtimali yüksektir. Mutasyon işlemiyle, genleri oluşturan bitler rastgele değiştirilerek bu olumsuzluk engellenir ve birey çeşitliliği artar.

Çalışmada, mutasyon metodlarından biri olan ters çevirme mutasyonu kullanılmış ve mutasyon oranı %5 olarak uygulanmıştır.

5.3.7. Yeni Popülasyonun oluşturulması ve en iyi bireyin seçilmesi

Başlangıç popülasyonuna çeşitli işlemlerin uygulanmasının ardından; anne, baba ve çocuk bireylerden oluşan popülasyon elde edilmiştir. Toplamda 200 bireyden oluşan bu popülasyonda tüm bireylerin uygunlukları hesaplanmış ve en yüksek uygunluk değerine sahip 100 birey bir sonraki nesle aktarılarak iyileştirme işlemi gerçekleştirilmiştir.

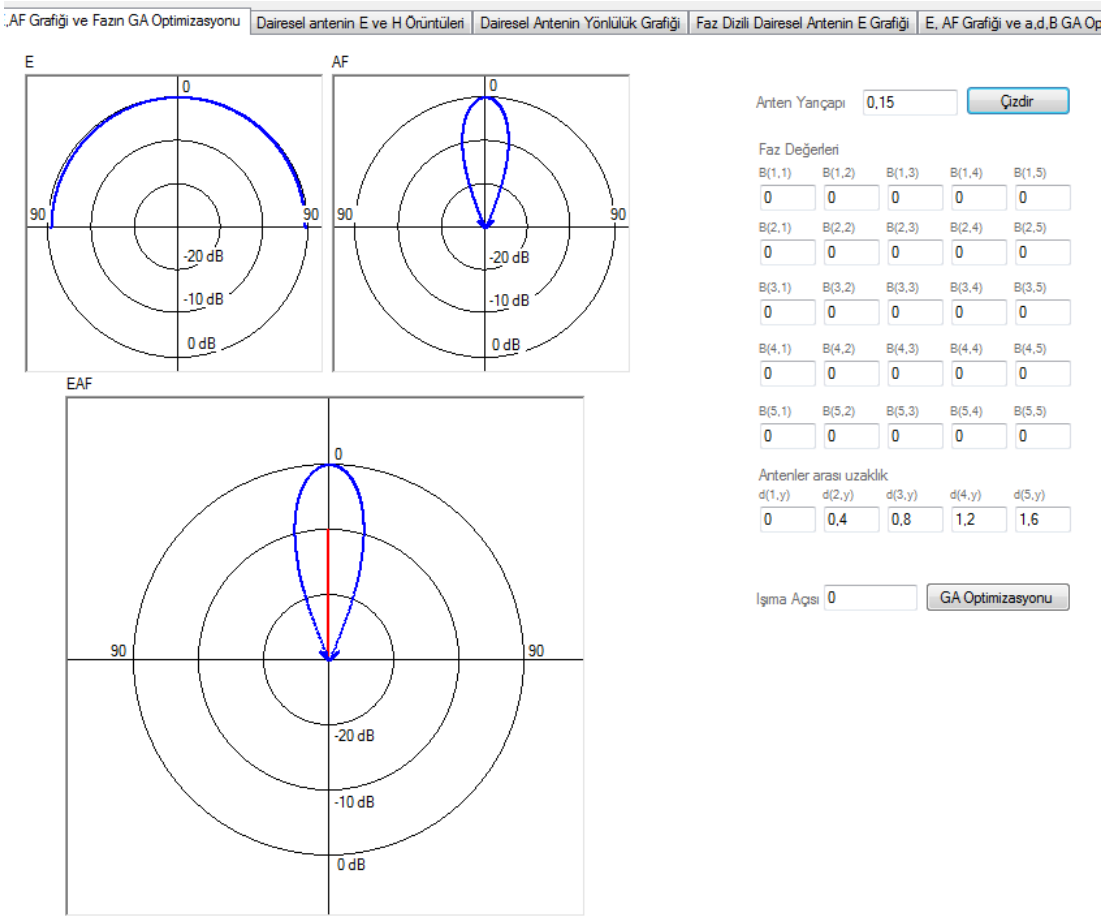
Seçim, çaprazlama, mutasyon, yeni popülasyon ve iyileştirme işlemleri için iterasyon sayısı 100 olarak belirlenmiştir. Her bir iterasyonda, bu işlemler tekrar edilmiştir. İterasyonun tamamlanmasının ardından elde edilen nesilde, en iyi uygunluk değerine sahip birey, problemin sonucunu vermektedir.

5.3.8. Problem çözümünün programlanması

Anten dizisinin modellenmesi ve istenen çıktıların genetik algoritma yoluyla elde edilmesinde kullanılan program, C# programla dilinde programlanmış ve grafik arayüzü ile sunulmuştur. Programın arayüz görünümü Şekil 5.11.'da gösterilmektedir.

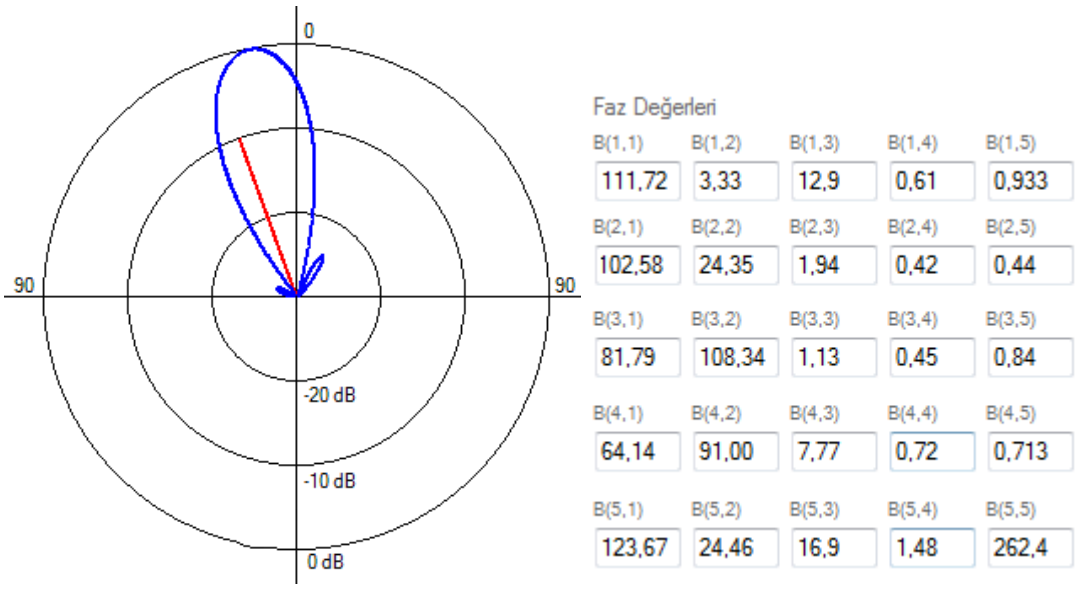
Görüleceği üzere, anten dizisinin parametreleri olan yarıçap, elemanlar arasındaki uzaklık ve her bir elemanın faz değerleri değiştirilebilmektedir. Yazı kutucuklarından alınan değerler kullanılarak anten elemanının, dizi çarpanının ve anten dizisinin ışıma örüntüleri butona basılarak grafiksel olarak elde edilebilmektedir. Çalışmada $\Phi = 0$ düzlemindeki ışıma örüntüleri elde edildiğinden, antenlerin x eksenine olan uzaklıklarının (d_y) etkisi yoktur. Bu nedenle elemanların yalnızca y düzlemine olan uzaklıkları (d_x) alınmaktadır.

Modellenen antenin genetik algoritma yoluyla elde edilebilmesi için, maksimum ışıma yapılması istenen açı değeri ve diğer anten parametreleri yazı kutucuklarından alınır. Genetik algoritma işleminin tamamlanmasının ardından elde edilen faz değeri sonuçları, yine yazı kutucuklarına yazdırılır. Çizdir butonuna basılarak, genetik algoritma işleminden dönen değerler ile elde edilen ışıma örüntüleri gözlemlenerek çözümün sağlaması yapılabilir.



Şekil 5.11. Çözüm Programının Arayüzü

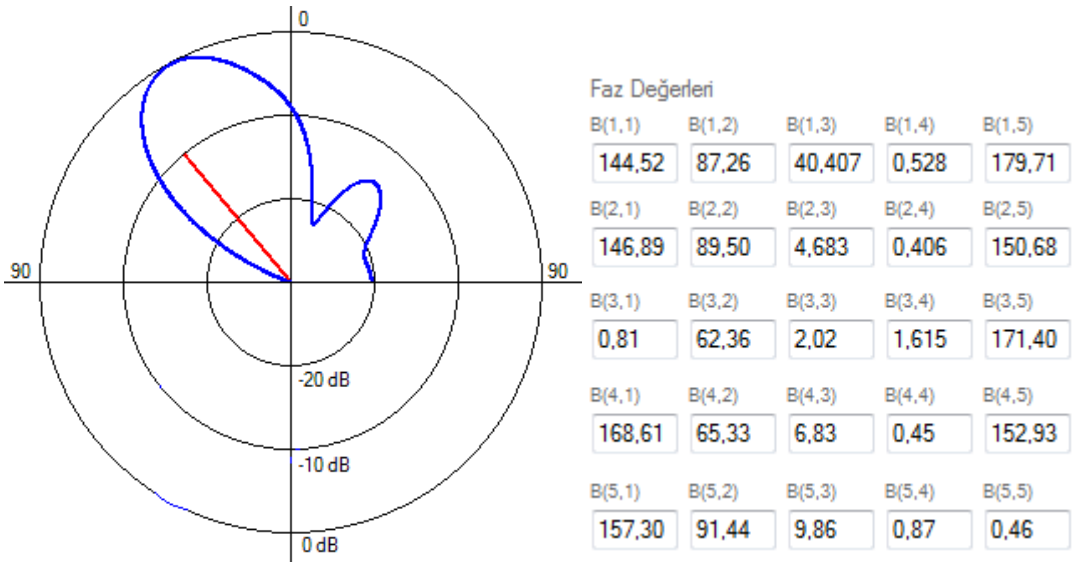
Programda ana hüzmenin yönlenmesi için istenen açı değerleri girilmiş, gerçekleştirilen optimizasyon işlemleri sonucu elde edilen faz değerleri ve bu faz değerleri sonucu oluşan ışıma örüntüleri Şekil 5.12, Şekil 5.13 ve Şekil 5.14'de verilmiştir.



Şekil 5.12. 20^0 için elde edilen ışıma örüntüsü ve faz değerleri

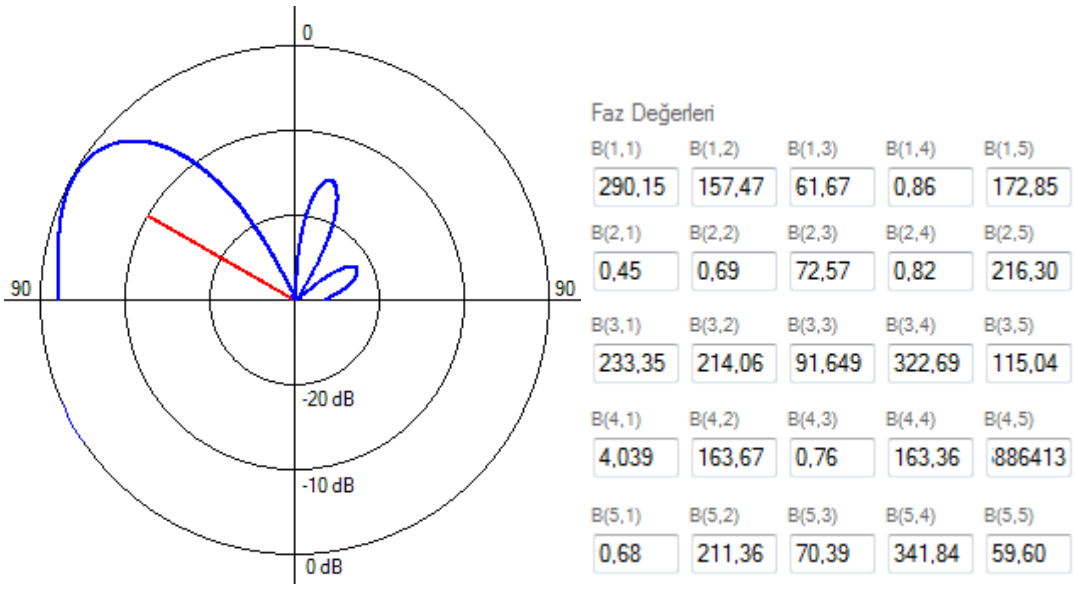
Şekil 5.12'de 20^0 doğrultusunda antenin maksimum yönlendirme yapması istenmiş ve bu değer için optimizasyon işlemi gerçekleştirilmiştir. Şekilde elde edilen ışıma örüntüsü ve bu örüntüyü verecek olan faz değerleri verilmektedir. Ana huzmenin tam olarak 20^0 doğrultusunda olmaması, optimizasyon işleminin en iyi sonucu bazen verememe ihtimalinin olduğunu göstermektedir. Optimizasyon işlemi gerçekleştirirken seçilen popülasyon büyüklüğü, iterasyon sayısının artırılması daha doğru sonuç verilmesini sağlar. Ancak bu değerler artırıldıkça programın hesaplama süresinin artacağı göz önünde bulundurulmalıdır.

Şekil 5.13'de, 40^0 için optimizasyon işlemi gerçekleştirilmiştir. Şekilde sunulan ışıma örüntüsünde Şekil 5.14'de olduğu gibi ana huzmenin istenen doğrultuda tam olarak yönlenemediği görülmektedir. Bunun sebebinin optimizasyon işleminde en doğru sonucun her zaman bulunma ihtimalinin olmadığı belirtilmiştir. İterasyon sayısı ve popülasyon büyüklüğünün artırılması seçeneğiyle birlikte, optimizasyon işleminin tekrar gerçekleştirilmesi ve istenen sonucun bulunana kadar tekrara devam edilmesi doğru sonucun bulunmasına yardımcı olur. 40^0 için elde edilen örüntüde ayrı ikinci bir yan lobun oluştuğu görülmektedir.



Şekil 5.13. 40^0 için elde edilen ışıma örüntüsü ve faz değerleri

Anten yönlendirmesinde yan lob sayısının artması ve büyümesi olumsuz karşılanmaktadır. Ancak ana huzme ile karşılaştırıldığında etkin büyüklüğe sahip olmayan yan loblar ihmal edilebilir.



Şekil 5.14. 60^0 için elde edilen ışıma örüntüsü ve faz değerleri

Şekil 5.14'de 60^0 için gerçekleştirilen optimizasyon işleminin sonuçları görülmektedir. 20^0 ve 40^0 için elde edilen sonuçların aksine, 60^0 için istenen

doğrultuda maksimum ışıma değerinin elde edildiği görülebilir. Bununla birlikte ana huzmenin dışında iki adet yan lob oluşmuştur. Ancak bu yan loblar ana huzmeyle karşılaştırıldığında ihmal edilebilecek seviyededir.

Teoride, anten dizisi yukarı ve alt kısma aynı şekilde yayılım yapmakta ve bu nedenle simetrik bir ışıma örüntüsü elde edilmektedir. Ancak pratikte, anten dizisinin alt tabakasının altında toprak yüzeyi bulunduğu için, alt kısma ışıma gerçekleşmez. Bu nedenle ışıma örüntüsünün alt kısmı gösterilmemiştir. 20^0, 40^0 ve 60^0 için optimizasyon işlemi tekrar denendiğinde farklı sonuçlar elde edilebilir. Bunun nedeni, optimizasyon işleminin kesin sonuç vermemesi, gerçekleştirdiği işlem adımlarıyla en iyiye yakın sonucu bulmaya çalışmasıdır. Bu nedenle tekrar yapılan bir optimizasyon işlemiyle en iyiye yakın farklı bir sonuç elde edilebilir. Bununla birlikte, çözüm programına girdi olarak verilen popülasyon sayısı ve iterasyon sayısı değerleri artırılarak en iyi sonuca ulaşma ihtimali artırılabilir. Fakat bu değerlerin artırılmasının, işlem süresini artıracak olmasına dikkat edilmelidir.

6. SONUÇ

Günümüzde, telekomünikasyon ve askeri uygulamalarda anten dizilerinin kullanımı gittikçe artmaktadır. Anten dizilerinin tercih edilmesinin en önemli sebebi, ışıma örüntüsünün donanımsal değişikliklere ihtiyaç duymadan, yazılımsal olarak değiştirilebilmesidir. İstenen çıktıların elde edilmesi için anten dizisi parametrelerinin sahip olması gereken değerlerin hesaplanması, önemli problemlerden biridir.

Bu çalışmada, anten dizilerinin maksimum ışıma hüzmesinin istenen doğrultuda olması için gereken faz parametrelerinin genetik algoritma yoluyla elde edilebileceği gösterilmiştir. Çalışmada C# programlama dili kullanılarak kullanım kolaylığı sağlayan arayüze sahip bir program geliştirilmiştir. Programda, anten dizisinin değişkenleri sabit alınarak, yalnızca faz değerlerinin hangi değerlere sahip olması gerektiği GA yoluyla çözümlenmiştir. Ancak program esnek olarak geliştirilmiş olup, ufak revizyonlarla farklı parametrelerin optimizasyonu için de kullanılabilir.

Modellenen anten dizisinde, özdeş mikroşerit dairesel antenler kullanılmıştır. Dairesel mikroşerit antenlerin anten dizileri oluşturulmasında sağladığı avantajlardan bahsedilmiştir. Optimizasyon işleminde, anten elemanlarının yapıları ve birbirleri arasındaki uzaklıklar sabit kabul edilmiş, elemanların yalnızca faz değerleri optimize edilmiştir. Bu sayede anten dizisinin fiziksel özelliklerinde değişiklik yapılmadan yalnızca besleme sinyali değiştirilerek istenen ışıma örüntüsü sağlanabilmektedir.

Çalışmanın devamı olarak kabul edilebilecek ileriki çalışmalarda, düzlemsel anten dizileri dışında kullanılan ve bazı uygulamalarda yer bulan silindirik ve küresel anten dizilerinin tasarımı için genetik algoritma ve benzer en iyileme metodları kullanarak optimizasyon çalışmaları yapılabilir.

KAYNAKLAR

1. Schelkunoff, S.A., “A mathematical theory of linear arrays”, ***Bell System Technical Journal,*** 22:80-147, (1943).

2. Balanis C. A., “Antenna theory analysis and design”, 2nd ed., ***John Wiley & Sons,*** New York, 204-282, (1982).

3. Woodward, P.M., “A method for calculating the field over a plane aperture required to produve a given polar diagram”, ***J. IEEE,*** 93:1554-1558, (1946).

4. Orchard, H. J., Elliot, R. S., Stern, J. G., “Optimizing the synthesis of shaped beam antenna patterns” ***IEE Proc. Microvave, Antennas Propagat.,*** 132:63-68, (1985).

5. Dolph, C. L., “A current distribution for broadside arrays which optimizes the relationship between beamwidth and sidelobe level” ***Proc. IRE Waves and Electrons,*** (1946).

6. Taylor, T.T., “Design of line source antennas for narrow beamwidth and low sidelobe”, ***IRE Trans. Antennas Propagat.,*** 3:16-28, (1955).

7. Deschamps, G. A., “Microstrip microwave antennas”, ***Proc. 3rd USAF Symposium on Antennas,*** (1953).

8. Howell, Q. E., “Microstrip antennas”, ***IEEE Trans. Antennas Propagat.,*** 23:90-93, (1975).

9. Pozar, D. M., “Microstrip antennas”, ***Proc. IEEE***, 80(1):79-81, (1992)

10. Yazgan, E., “Mikroşerit antenler”, ***Elektrik Mühendisliği Dergisi,*** 348:262-268, (1987).

11. Başaran, C. “Kablosuz haberleşme uygulamaları için yarık-halka mikroşerit anten tasarımı”, ***Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi,*** Kocaeli, (2008).

12. Garg, R., Bhartia, P., Bahl, I., Ittipiboon, A., “Microstrip antenna design handbook”, ***Artech House Antennas and Propagation Library,*** 1-2, (2001).

13. Pozar, D., “An update on microstrip antenna theory and design including some novel feeding techniques”, ***Antennas and Propagation Society Newsletter, IEEE,*** 28(5):4-9, (1986).

14. Fong, K. S., Pues, H.F., Withers, M. J., “Wideband multilayer coaxial-fed microstrip antenna element”, ***Electronic Letters,*** 21(11):497-499, (1985).

15. Pozar, D. M., Kaufman, B., “Increasing the bandwidth of a microstrip antenna by proximity coupling”, ***Electronic Letters,*** 23(8):368-369, (1987).

16. Pozar, D. M., Kaufmann, B., “Increasing the bandwidth of a microstrip antenna by proximity coupling”, ***Electronic Letters,*** 23:368-369, (1987).

17. Targonski, S. D., Waterhouse, R. B., Pozar, D. M., “Design of wide-band aperture stacked patch microstrip antennas”, ***IEEE Trans. On Antennas and Propagation,*** 46:1245-1251, (1998).

18. Balanis, C.A., “Advanced engineering electromagnetics”, ***John Wiley and Sons,*** New York (1989).

19. Watkins, J., “Circular resonant structures in microstrip”, ***Electronic Letters,*** 5(21): 524-525, (1969).

20. Pozar, D.M., Schaubert, D. H., “Microstrip antennas", ***IEEE Press,*** New York, (1995).

21. Aksoy, E., Afacan, E., “Planar antenna pattern nulling using differential evolution algorithm”, ***International Journal of Electronics and Communications,*** 63(2):116-122, (2009).

22. Holland, J. H. “Adaptation in natural and artificial systems”, ***Ann Arbor,*** University of Michigan Press, (1975).

23. Goldberg, D.E., “Genetic algorithms in search, optimization, and machine Learning”, ***Reading, MA*** Addison-Wesley, (1989).

24. Haupt, R.L., Haupt, S.E., “Practical genetic algorithms”, ***John Wiley & Sons,*** New Jersey, (2004).

25. Qu, L., Sun, R., “A synergetic approach to genetic algorithms for solving traveling salesman problem”, ***Information Sciences,*** 117:267-283, (1999).

26. De Jong, K.A., “An analysis of the behaviour of a class of genetic adaptive systems”, ***PhD Dissertation,*** University of Michigan, (1975).

27. Melanie, M., “An introduction to genetic algorithms”, ***A Bradford Book the MIT Press,*** London, (1998).

28. Emel, G., Taşkın, Ç., "Genetik algoritmalar ve uygulama alanları", ***Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi,*** 129-152, (2002).

29. Gen, M., Cheng, R., "Genetic algorithms and engineering design", ***John Wiley & Sons***, Canada, (1997).

30. Kurt, M., Semetay, C., "Genetik algoritma ve uygulama alanları", ***M.M.O. Makine Mühendis Dergisi***, 42(501):19-24, (2001).

31. Turgut, P., Arslan, A., "Sürekli bir kirişte maksimum momentlerin genetik algoritmalar ile belirlenmesi", ***DEÜ Fen ve Mühendislik Dergisi***, 3:1-9, (2001).

32. Cengiz, Y., "Optimum performanslı mikrodalga kuvvetlendirici tasarımı", ***Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi***, İstanbul, (2004).

33. Brockus, C. G., "Shortest path optimization using a genetics search technique, modeling and simulation", ***Proc of the 14th Annual Pittsburgh Conference,*** 14:241-245, (1983).

34. Booker, L. B., Goldberg, D. E., Holland, J. H., "Classifier system and genetic algorithms", ***Artificial Intelligence,*** 40:235-282, (1989).

35. Mansfield, R. A., "Genetic algorithms", ***University of Wales College of Cardiff, Master Thesis,*** Cardiff, U.K., (1990).

EK – 1. Çözüm Programının Kodları

```
//Çizim butonu
private void button13_Click(object sender, EventArgs e)
      {
         double i = 0;      //döngü değişkeni
         double teta;       //teta açısı
         double ae = Convert.ToDouble(textBox13.Text);
                  //yarıçap değerini al
         k0 = 2 * Math.PI * f0 / c; //k0 değerini tanımla
         int kalinlik = 1; //grafik için çizgi kalınlığı
         double E_disdaire = 100, AF_disdaire = 100, EAF_disdaire = 150;
                  //grafik için daireler

         double E = 0, E_sin, E_cos;       //Antenin elektrik alanı
         double AF = 0, AF_sin, AF_cos; //Dizi faktörü
         double EAF = 0, EAF_sin, EAF_cos;
                                    //Anten dizisinin elektrik alanı
         double E_norm = 0, AF_norm = 0, EAF_norm = 0;
                                    //normalizasyon değerleri
         double E_max = 0, AF_max = 0, EAF_max = 0;
                                 //maksimum değerler
         double[] b = new double[25];      //faz değerleri
         double[] d = new double[25]; //antenler arası uzaklıklar

                  //faz değerlerini al
         b[0] = Convert.ToDouble(textBox110.Text);
         b[1] = Convert.ToDouble(textBox111.Text);
         b[2] = Convert.ToDouble(textBox112.Text);
         b[3] = Convert.ToDouble(textBox113.Text);
         b[4] = Convert.ToDouble(textBox114.Text);
         b[5] = Convert.ToDouble(textBox115.Text);
         b[6] = Convert.ToDouble(textBox116.Text);
         b[7] = Convert.ToDouble(textBox117.Text);
         b[8] = Convert.ToDouble(textBox118.Text);
         b[9] = Convert.ToDouble(textBox119.Text);
         b[10] = Convert.ToDouble(textBox120.Text);
         b[11] = Convert.ToDouble(textBox121.Text);
         b[12] = Convert.ToDouble(textBox122.Text);
         b[13] = Convert.ToDouble(textBox123.Text);
         b[14] = Convert.ToDouble(textBox124.Text);
         b[15] = Convert.ToDouble(textBox125.Text);
         b[16] = Convert.ToDouble(textBox126.Text);
         b[17] = Convert.ToDouble(textBox127.Text);
         b[18] = Convert.ToDouble(textBox128.Text);
         b[19] = Convert.ToDouble(textBox129.Text);
         b[20] = Convert.ToDouble(textBox130.Text);
         b[21] = Convert.ToDouble(textBox131.Text);
         b[22] = Convert.ToDouble(textBox132.Text);
         b[23] = Convert.ToDouble(textBox133.Text);
         b[24] = Convert.ToDouble(textBox134.Text);

         //antenler arasındaki uzaklık değerlerini al
         d[0] = Convert.ToDouble(textBox210.Text);
         d[1] = Convert.ToDouble(textBox211.Text);
```

EK – 1 (Devam). Çözüm Programının Kodları

```
 d[2] = Convert.ToDouble(textBox212.Text);
d[3] = Convert.ToDouble(textBox213.Text);
d[4] = Convert.ToDouble(textBox214.Text);
d[5] = Convert.ToDouble(textBox210.Text);
d[6] = Convert.ToDouble(textBox211.Text);
d[7] = Convert.ToDouble(textBox212.Text);
d[8] = Convert.ToDouble(textBox213.Text);
d[9] = Convert.ToDouble(textBox214.Text);
d[10] = Convert.ToDouble(textBox210.Text);
d[11] = Convert.ToDouble(textBox211.Text);
d[12] = Convert.ToDouble(textBox212.Text);
d[13] = Convert.ToDouble(textBox213.Text);
d[14] = Convert.ToDouble(textBox214.Text);
d[15] = Convert.ToDouble(textBox210.Text);
d[16] = Convert.ToDouble(textBox211.Text);
d[17] = Convert.ToDouble(textBox212.Text);
d[18] = Convert.ToDouble(textBox213.Text);
d[19] = Convert.ToDouble(textBox214.Text);
d[20] = Convert.ToDouble(textBox210.Text);
d[21] = Convert.ToDouble(textBox211.Text);
d[22] = Convert.ToDouble(textBox212.Text);
d[23] = Convert.ToDouble(textBox213.Text);
d[24] = Convert.ToDouble(textBox214.Text);

panel9.Refresh();
panel10.Refresh();
panel11.Refresh();
listBox2.Items.Clear();

Graphics g4 = this.panel9.CreateGraphics();
          //E için grafik fonksiyonu
Graphics g5 = this.panel10.CreateGraphics();
          //AF için grafik fonksiyonu
Graphics g6 = this.panel11.CreateGraphics();
          //EAF için grafik fonksiyonu

Pen kalem = new Pen(Color.Blue, 2);
          //ışıma örüntüsü için
Pen kalem2 = new Pen(Color.Black, 1);
          //eksenler ve daireler için
Pen kalem4 = new Pen(Color.Red, 2);
                                    // max çizgisi için
           Point yatay_eksen1 = new Point(0, panel9.Height / 2);                    //E
           için eksen çizgileri
Point yatay_eksen2 = new Point(panel9.Width, panel9.Height / 2);
Point dusey_eksen1 = new Point(panel9.Width / 2, 0);
Point dusey_eksen2 = new Point(panel9.Width / 2, panel9.Height);

Point yatay_eksen3 = new Point(0, panel10.Height / 2);
          //AF için eksen çizgileri
Point yatay_eksen4 = new Point(panel10.Width, panel10.Height / 2);
Point dusey_eksen3 = new Point(panel10.Width / 2, 0);
Point dusey_eksen4 = new Point(panel10.Width / 2, panel10.Height);
```

EK – 1 (Devam). Çözüm Programının Kodları

```
            Point yatay_eksen5 = new Point(0, panel11.Height / 2);
                        //EAF için eksen çizgileri
             Point yatay_eksen6 = new Point(panel11.Width, panel11.Height / 2);
             Point dusey_eksen5 = new Point(panel11.Width / 2, 00);
             Point dusey_eksen6 = new Point(panel11.Width / 2, panel11.Height);
             Point merkez_x3 = new Point(panel11.Width / 2, panel11.Height / 2);
              //merkez noktası

            g4.DrawLine(kalem2, yatay_eksen1, yatay_eksen2);
                                    //E için eksenleri çiz
            g4.DrawLine(kalem2, dusey_eksen1, dusey_eksen2);
            g4.DrawEllipse(kalem2, (float)(panel9.Width / 2 - E_disdaire), (float)(panel9.Height /
2 - E_disdaire), (int)E_disdaire * 2, (int)E_disdaire * 2);
                                          //E için daireleri çiz
            g4.DrawEllipse(kalem2, (float)(panel9.Width / 2 - E_disdaire / 3 * 2),
(float)(panel9.Height / 2 - E_disdaire / 3 * 2), (int)E_disdaire * 2 / 3 * 2, (int)E_disdaire * 2 / 3
* 2);
            g4.DrawEllipse(kalem2, (float)(panel9.Width / 2 - E_disdaire / 3),
(float)(panel9.Height / 2 - E_disdaire / 3), (int)E_disdaire * 2 / 3, (int)E_disdaire * 2 / 3);


            g5.DrawLine(kalem2, yatay_eksen3, yatay_eksen4);
                        //AF için eksenleri çiz
            g5.DrawLine(kalem2, dusey_eksen3, dusey_eksen4);
            g5.DrawEllipse(kalem2, (float)(panel10.Width / 2 - AF_disdaire),
(float)(panel10.Height / 2 - AF_disdaire), (int)AF_disdaire * 2, (int)AF_disdaire * 2); //AF için
daireleri çiz
            g5.DrawEllipse(kalem2, (float)(panel10.Width / 2 - AF_disdaire / 3 * 2),
(float)(panel10.Height / 2 - AF_disdaire / 3 * 2), (int)AF_disdaire * 2 / 3 * 2, (int)AF_disdaire *
2 / 3 * 2);
            g5.DrawEllipse(kalem2, (float)(panel10.Width / 2 - AF_disdaire / 3),
(float)(panel10.Height / 2 - AF_disdaire / 3), (int)AF_disdaire * 2 / 3, (int)AF_disdaire * 2 / 3);

            g6.DrawLine(kalem2, yatay_eksen5, yatay_eksen6); //EAF için eksenleri çiz
            g6.DrawLine(kalem2, dusey_eksen5, dusey_eksen6);
            g6.DrawEllipse(kalem2, (float)(panel11.Width / 2 - EAF_disdaire),
(float)(panel11.Height / 2 - EAF_disdaire), (int)EAF_disdaire * 2, (int)EAF_disdaire * 2);
//EAF için daireleri çiz
            g6.DrawEllipse(kalem2, (float)(panel11.Width / 2 - EAF_disdaire / 3 * 2),
(float)(panel11.Height / 2 - EAF_disdaire / 3 * 2), (int)EAF_disdaire * 2 / 3 * 2,
(int)EAF_disdaire * 2 / 3 * 2);
            g6.DrawEllipse(kalem2, (float)(panel11.Width / 2 - EAF_disdaire / 3),
(float)(panel11.Height / 2 - EAF_disdaire / 3), (int)EAF_disdaire * 2 / 3, (int)EAF_disdaire * 2
/ 3);



            Point max_ang = new Point(panel11.Width / 2 + Convert.ToInt32(100 *
Math.Cos((Convert.ToInt32(textBox_maxang.Text)+90) * Math.PI / 180)), panel11.Height / 2
- Convert.ToInt32(100 * Math.Sin((Convert.ToInt32(textBox_maxang.Text)+90) * Math.PI /
180)));   //doğrultu açısı noktası

            g6.DrawLine(kalem4, merkez_x3, max_ang);                    //doğrultu çizgisini çiz
```

EK – 1 (Devam). Çözüm Programının Kodları

```
//maksimum hesaplama işlemleri
for (i = 0; i < 360; i = i + 0.1)
{
    teta = i * Math.PI / 180;

    E = E_Hesapla(teta, ae);
    if (E > E_max) E_max = E;

    AF = AF_hesapla(d, b, teta);
    if (AF > AF_max) AF_max = AF;

    EAF = EAF_Hesapla2(i, ae, d, b);
    if (EAF > EAF_max) EAF_max = EAF;
}

//normalizasyon işlemleri
for (i = 0; i < 360; i = i + 0.1)
{
    teta = i * Math.PI / 180;

    if (E_max > 0)
    {
        E_norm = 20 * Math.Log(Math.Abs(E_Hesapla(teta, ae)) / E_max);
        if (E_norm == 0)
        {
            E = 1;

        }
        if(E_norm<-30)
            E=0;
        else
            E = (E_norm+30)/3;


    }
    else
        E = 1;


    if (AF_max > 0)
    {
        AF_norm = 20 * Math.Log(Math.Abs(AF_hesapla(d, b, teta)) / AF_max);

        if (AF_norm == 0)
        {
            AF = 1;

        }
        if(AF_norm<-30)
            AF=0;
        else
            AF = (AF_norm+30)/3;
```

EK – 1 (Devam). Çözüm Programının Kodları

```
            }
            else
               AF = 1;


             if (EAF_max > 0)
            {
               EAF_norm = 20 * Math.Log(Math.Abs(EAF_Hesapla2(i, ae, d, b)) / EAF_max);

               if (EAF_norm == 0)
               {
                  EAF = 1;

               }
               if(EAF_norm<-30)
                  EAF=0;
               else
                  EAF = (EAF_norm+30)/3;


            }
            else
               EAF = 1;

            listBox2.Items.Add(Convert.ToString(Math.Round(i, 2)) + "- " +
Convert.ToString(EAF));

            E_sin = E * E_disdaire / 10 * Math.Sin(teta + 90 * Math.PI / 180);
            E_cos = E * E_disdaire / 10 * Math.Cos(teta + 90 * Math.PI / 180);

            AF_sin = AF * AF_disdaire / 10 * Math.Sin(teta + 90 * Math.PI / 180);
            AF_cos = AF * AF_disdaire / 10 * Math.Cos(teta + 90 * Math.PI / 180);

            EAF_sin = EAF * EAF_disdaire / 10 * Math.Sin(teta + 90 * Math.PI / 180);
            EAF_cos = EAF * EAF_disdaire / 10 * Math.Cos(teta + 90 * Math.PI / 180);

            //ışıma örüntülerini çizdir:
            g4.DrawLine(kalem, panel9.Width / 2 + Convert.ToInt32(E_cos), panel9.Height / 2
- Convert.ToInt32(E_sin), panel9.Width / 2 + Convert.ToInt32(E_cos) + kalinlik,
panel9.Height / 2 - Convert.ToInt32(E_sin) + kalinlik);
            g5.DrawLine(kalem, panel10.Width / 2 + Convert.ToInt32(AF_cos), panel10.Height
/ 2 - Convert.ToInt32(AF_sin), panel10.Width / 2 + Convert.ToInt32(AF_cos) + kalinlik,
panel10.Height / 2 - Convert.ToInt32(AF_sin) + kalinlik);
            g6.DrawLine(kalem, panel11.Width / 2 + Convert.ToInt32(EAF_cos),
panel11.Height / 2 - Convert.ToInt32(EAF_sin), panel11.Width / 2 +
Convert.ToInt32(EAF_cos) + kalinlik, panel11.Height / 2 - Convert.ToInt32(EAF_sin) +
kalinlik);

         }

      }


//  Optimizasyon butonu
```

EK – 1 (Devam). Çözüm Programının Kodları

```
  private void button14_Click(object sender, EventArgs e)
      {

          k0 = 2 * Math.PI * f0 / c;

          //  Çaprazlama oranı                = 80%
          //  Mutasyon                        =  5%
          //  Populasyon büyüklüğü    = 100
          //  Nesil sayısı (iterasyon)= 200
          //  Kromozom büyüklüğü              = 25

          GA ga = new GA(0.8, 0.05, 100, 200, 25);                // ilk popülasyonun yaratılması
ve crossover-mutasyon vs. değerlerinin atanması

          ga.FitnessFunction = new GAFunction(theActualFunction2);    //fitness
fonksiyonunun belirlenmesi

          ga.Elitism = true;                                  //ilk değer atamaları
          ga.Go();                                          //işlemi gerçekleştir

          double[] values;
          double fitness;
          ga.GetBest(out values, out fitness);                  //en iyi bireyi al


          //Yazı kutularına değerleri yaz:
          textBox110.Text = Convert.ToString(values[0]);
          textBox111.Text = Convert.ToString(values[1]);
          textBox112.Text = Convert.ToString(values[2]);
          textBox113.Text = Convert.ToString(values[3]);
          textBox114.Text = Convert.ToString(values[4]);
          textBox115.Text = Convert.ToString(values[5]);
          textBox116.Text = Convert.ToString(values[6]);
          textBox117.Text = Convert.ToString(values[7]);
          textBox118.Text = Convert.ToString(values[8]);
          textBox119.Text = Convert.ToString(values[9]);
          textBox120.Text = Convert.ToString(values[10]);
          textBox121.Text = Convert.ToString(values[11]);
          textBox122.Text = Convert.ToString(values[12]);
          textBox123.Text = Convert.ToString(values[13]);
          textBox124.Text = Convert.ToString(values[14]);
          textBox125.Text = Convert.ToString(values[15]);
          textBox126.Text = Convert.ToString(values[16]);
          textBox127.Text = Convert.ToString(values[17]);
          textBox128.Text = Convert.ToString(values[18]);
          textBox129.Text = Convert.ToString(values[19]);
          textBox130.Text = Convert.ToString(values[20]);
          textBox131.Text = Convert.ToString(values[21]);
          textBox132.Text = Convert.ToString(values[22]);
          textBox133.Text = Convert.ToString(values[23]);
          textBox134.Text = Convert.ToString(values[24]);

          //Konsola yazdır:
          System.Console.WriteLine("Best ({0}):", fitness);
```

EK – 1 (Devam). Çözüm Programının Kodları

```
        for (int i = 0; i < values.Length; i++)
            System.Console.WriteLine("{0} ", values[i]);


        ga.GetWorst(out values, out fitness);
        System.Console.WriteLine("\nWorst ({0}):", fitness);
        for (int i = 0; i < values.Length; i++)
            System.Console.WriteLine("{0} ", values[i]);

    }



    private Double EAF_Hesapla2(Double teta, Double ae, Double[] d, Double[] b)
    {

        teta = teta * Math.PI / 180;

        Double Eteta = E_Hesapla(teta, ae);

        Double AF = 0;

        AF = AF_hesapla(d,b, teta);


        return Math.Abs(AF) * Math.Abs(Eteta);
    }

    private Double E_Hesapla(double teta, double ae)
    {
        Double Eteta = k0 * ae * v0 / (2 * r) * j02i(k0 * ae * System.Math.Sin(teta));
        return Math.Abs( Eteta);

    }

    private Double AF_hesapla(double[] d, double[] b, double teta)
    {

        int n = Convert.ToInt32(b.Length);

        double AF=0, AF_reel=0, AF_img=0;

        for (int i = 0; i < n; i++)
        {
            AF_reel= AF_reel + Math.Cos(b[i]*Math.PI/180 + 2*Math.PI*d[i]*Math.Sin(teta));
            AF_img= AF_img + Math.Sin(b[i]*Math.PI/180 + 2*Math.PI*d[i]*Math.Sin(teta));
        }

        AF = Math.Sqrt(AF_reel * AF_reel + AF_img * AF_img);

        return AF;


    }
```

EK – 1 (Devam). Çözüm Programının Kodları

```
 private double theActualFunction2(double[] values)
     {
         if (values.GetLength(0) != 25)
            throw new ArgumentOutOfRangeException("should have 25 args");

         //Double tetaMin = Convert.ToDouble(textBox15.Text);
         Double tetaMax = Convert.ToDouble(textBox_maxang.Text);

         double toplam = 0, ortalama = 0, ae;
         ae = Convert.ToDouble(textBox13.Text);
         double[] d = new double[25];


         d[0] = Convert.ToDouble(textBox210.Text);
         d[1] = Convert.ToDouble(textBox211.Text);
         d[2] = Convert.ToDouble(textBox212.Text);
         d[3] = Convert.ToDouble(textBox213.Text);
         d[4] = Convert.ToDouble(textBox214.Text);
         d[5] = Convert.ToDouble(textBox210.Text);
         d[6] = Convert.ToDouble(textBox211.Text);
         d[7] = Convert.ToDouble(textBox212.Text);
         d[8] = Convert.ToDouble(textBox213.Text);
         d[9] = Convert.ToDouble(textBox214.Text);
         d[10] = Convert.ToDouble(textBox210.Text);
         d[11] = Convert.ToDouble(textBox211.Text);
         d[12] = Convert.ToDouble(textBox212.Text);
         d[13] = Convert.ToDouble(textBox213.Text);
         d[14] = Convert.ToDouble(textBox214.Text);
         d[15] = Convert.ToDouble(textBox210.Text);
         d[16] = Convert.ToDouble(textBox211.Text);
         d[17] = Convert.ToDouble(textBox212.Text);
         d[18] = Convert.ToDouble(textBox213.Text);
         d[19] = Convert.ToDouble(textBox214.Text);
         d[20] = Convert.ToDouble(textBox210.Text);
         d[21] = Convert.ToDouble(textBox211.Text);
         d[22] = Convert.ToDouble(textBox212.Text);
         d[23] = Convert.ToDouble(textBox213.Text);
         d[24] = Convert.ToDouble(textBox214.Text);


         for (int i = 0; i < 360; i = i + 5)
         {
            toplam = toplam + EAF_Hesapla2(i, ae, d, values);
         }
         ortalama = toplam / 72;

         double max_EAF = EAF_Hesapla2(tetaMax, ae, d, values);
         //double EAF_Sapma1 = EAF_Hesapla(tetaMax+10, values[0], values[1], values[3],
values[2]);
         //double EAF_Sapma2 = EAF_Hesapla(tetaMax - 10, values[0], values[1], values[3],
values[2]);
         //double mainlobe = (3*max_EAF + EAF_Sapma1 + EAF_Sapma2) / 3;
```

EK – 1 (Devam). Çözüm Programının Kodları

```
            Double f1 = (max_EAF) - ortalama;    /* +(1/32))  / (EAF_Hesapla(tetaMin, values[0],
values[1], values[3], values[2]) + (1 / 32) ) */ ;
            return f1;
        }




            public delegate double GAFunction(double[] values);

            /// <summary>
            /// Genetic Algoritma class'ı
            /// </summary>
            public class GA
            {
                        /// <summary>
                        /// Varsayılan olarak mutasyon oranı %5, çaprazlama %80,
        /// popülasyon 100 ve nesil sayısı 2000'dir
                        /// </summary>
                        public GA()
                        {
                                    InitialValues();
                                    m_mutationRate = 0.05;
                                    m_crossoverRate = 0.80;
                                    m_populationSize = 100;
                                    m_generationSize = 2000;
                                    m_strFitness = "";
                        }

                        public GA(double crossoverRate, double mutationRate, int populationSize,
int generationSize, int genomeSize)
                        {
                                    InitialValues();
                                    m_mutationRate = mutationRate;
                                    m_crossoverRate = crossoverRate;
                                    m_populationSize = populationSize;
                                    m_generationSize = generationSize;
                                    m_genomeSize = genomeSize;
                                    m_strFitness = "";
                        }

                        public GA(int genomeSize)
                        {
                                    InitialValues();
                                    m_genomeSize = genomeSize;
                        }


                        public void InitialValues()
                        {
                                    m_elitism = false;
                        }
```

EK – 1 (Devam). Çözüm Programının Kodları

```
        /// <summary>
        /// GA işlemini başlatan fonksiyon
        /// </summary>
        public void Go()
        {
            if (getFitness == null)
                throw new ArgumentNullException("Need to supply fitness
function");
            if (m_genomeSize == 0)
                throw new IndexOutOfRangeException("Genome size not
set");

            //  Uygunluk tablosunu oluştur
            m_fitnessTable = new ArrayList();
            m_thisGeneration = new ArrayList(m_generationSize);
            m_nextGeneration = new ArrayList(m_generationSize);
            Genome.MutationRate = m_mutationRate;


            CreateGenomes();        //Kromozomları Oluştur
            RankPopulation();       //Popülasyondaki bireyleri değerlendir

            StreamWriter outputFitness = null;

            for (int i = 0; i < m_generationSize; i++)
            {
                CreateNextGeneration();     //yeni nesle aktaracak bireyleri
oluştur
                RankPopulation();           //o bireyleri değerlendirip sırala

            }
            if (outputFitness != null)
                outputFitness.Close();
        }

        /// <summary>
        /// Bütün kromozomları uyum fonksiyonuna göre sıraladıktan sonra, rulet
tekeri
    /// metodu kullanılır. Böylece yüksek uygunluğa sahip bireylerin daha fazla
    /// şansı vardır.
        /// </summary>
        /// <returns>En yüksek uyum değerine sahip bireyi döndürür</returns>
        private int RouletteSelection()
        {
            double randomFitness = m_random.NextDouble() * m_totalFitness;
            int idx = -1;
            int mid;
            int first = 0;
            int last = m_populationSize -1;
            mid = (last - first)/2;

            while (idx == -1 && first <= last)
            {
                if (randomFitness < (double)m_fitnessTable[mid])
```

EK – 1 (Devam). Çözüm Programının Kodları

```
                                {
                                        last = mid;
                                }
                                else if (randomFitness > (double)m_fitnessTable[mid])
                                {
                                        first = mid;
                                }
                                mid = (first + last)/2;
                                // i ve i+1 arasında
                                if ((last - first) == 1)
                                        idx = last;
                        }
                        return idx;
                }

                /// <summary>
                /// Popülasyonu derecelendir ve uyum fonksiyonuna göre sırala
                /// </summary>
                private void RankPopulation()
                {
                        m_totalFitness = 0;
                        for (int i = 0; i < m_populationSize; i++)
                        {
                                Genome g = ((Genome) m_thisGeneration[i]);
                                g.Fitness = FitnessFunction(g.Genes());
                                m_totalFitness += g.Fitness;
                        }

        m_thisGeneration.Sort(new GenomeComparer());

                        //  uyum değerine göre sıralar
                        double fitness = 0.0;
                        m_fitnessTable.Clear();
                        for (int i = 0; i < m_populationSize; i++)
                        {
                                fitness += ((Genome)m_thisGeneration[i]).Fitness;
                                m_fitnessTable.Add((double)fitness);
                        }
                }

                /// <summary>
                /// Başlangıç kromozomlarını oluştur
                /// </summary>
                private void CreateGenomes()
                {
                        for (int i = 0; i < m_populationSize ; i++)
                        {
                                Genome g = new Genome(m_genomeSize);
          m_thisGeneration.Add(g);          //yeni bir kromozom üretip jenerasyona ekle
                        }
                }

                private void CreateNextGeneration()        //bir sonraki nesle aktarılacak
bireyleri yarat
```

EK – 1 (Devam). Çözüm Programının Kodları

```
        {
            m_nextGeneration.Clear();
            Genome g = null;
            if (m_elitism)
                g = (Genome)m_thisGeneration[m_populationSize - 1];

            for (int i = 0 ; i < m_populationSize ; i+=2)
            {
                int pidx1 = RouletteSelection();          //rulet yöntemiyle
iyi iki bireyi seç
                int pidx2 = RouletteSelection();
                Genome parent1, parent2, child1, child2;
                parent1 = ((Genome) m_thisGeneration[pidx1]);
                parent2 = ((Genome) m_thisGeneration[pidx2]);

                if (m_random.NextDouble() < m_crossoverRate)
                {
                    parent1.Crossover(ref parent2, out child1, out
child2); //bireyleri çarpazla
                }
                else
                {
                    child1 = parent1;
                    child2 = parent2;
                }
                child1.Mutate();                  //mutasyona uğrat
                child2.Mutate();

                m_nextGeneration.Add(child1);          //yeni bireyleri
jenerasyona ekle
                m_nextGeneration.Add(child2);
            }
            if (m_elitism && g != null)
                m_nextGeneration[0] = g;              //en iyi bireyi taşı

            m_thisGeneration.Clear();
            for (int i = 0 ; i < m_populationSize; i++)
                m_thisGeneration.Add(m_nextGeneration[i]);
//popülasyon sayısı kadar yeni oluşturduğu bireylerden ekle
        }


        private double m_mutationRate;
        private double m_crossoverRate;
        private int m_populationSize;
        private int m_generationSize;
        private int m_genomeSize;
        private double m_totalFitness;
        private string m_strFitness;
        private bool m_elitism;

        private ArrayList m_thisGeneration;
        private ArrayList m_nextGeneration;
        private ArrayList m_fitnessTable;
```

EK – 1 (Devam). Çözüm Programının Kodları

```
static Random m_random = new Random();


static private GAFunction getFitness;
public GAFunction FitnessFunction
{
        get
        {
                return getFitness;
        }
        set
        {
                getFitness = value;
        }
}


//  Properties
public int PopulationSize
{
        get
        {
                return m_populationSize;
        }
        set
        {
                m_populationSize = value;
        }
}

public int Generations
{
        get
        {
                return m_generationSize;
        }
        set
        {
                m_generationSize = value;
        }
}

public int GenomeSize
{
        get
        {
                return m_genomeSize;
        }
        set
        {
                m_genomeSize = value;
        }
```

EK – 1 (Devam). Çözüm Programının Kodları

```
        }

        public double CrossoverRate
        {
                get
                {
                        return m_crossoverRate;
                }
                set
                {
                        m_crossoverRate = value;
                }
        }
        public double MutationRate
        {
                get
                {
                        return m_mutationRate;
                }
                set
                {
                        m_mutationRate = value;
                }
        }

        public string FitnessFile
        {
                get
                {
                        return m_strFitness;
                }
                set
                {
                        m_strFitness = value;
                }
        }

        /// <summary>
        /// Bir önceki neslin en iyi bireyini en kötüsünün yerine koyarak koru
        /// </summary>
        public bool Elitism
        {
                get
                {
                        return m_elitism;
                }
                set
                {
                        m_elitism = value;
                }
        }

        public void GetBest(out double[] values, out double fitness)       //en iyi bireyi
al
```

EK – 1 (Devam). Çözüm Programının Kodları

```
		{
			Genome g = ((Genome)m_thisGeneration[m_populationSize-1]);
			values = new double[g.Length];
			g.GetValues(ref values);
			fitness = (double)g.Fitness;
		}

		public void GetWorst(out double[] values, out double fitness)
		{
			GetNthGenome(0, out values, out fitness);
		}

		public void GetNthGenome(int n, out double[] values, out double fitness)
		{
			if (n < 0 || n > m_populationSize-1)
				throw new ArgumentOutOfRangeException("n çok büyük, ya
da çok küçük");
			Genome g = ((Genome)m_thisGeneration[n]);
			values = new double[g.Length];
			g.GetValues(ref values);
			fitness = (double)g.Fitness;
	}


//Genome class'ı
		public Genome(int length)
		{
			m_length = length;
			m_genes = new double[ length ];
			CreateGenes();
		}
		public Genome(int length, bool createGenes)
		{
			m_length = length;
			m_genes = new double[ length ];
			if (createGenes)
				CreateGenes();
		}

		public Genome(ref double[] genes)
		{
			m_length = genes.GetLength(0);
			m_genes = new double[ m_length ];
			for (int i = 0 ; i < m_length ; i++)
				m_genes[i] = genes[i];
		}


		private void CreateGenes()
		{
	//DateTime d = DateTime.UtcNow;
	for (int i = 0; i < m_length; i++)
	   m_genes[i] = m_random.NextDouble() * 360;
```

EK – 1 (Devam). Çözüm Programının Kodları

```
        }

        public void Crossover(ref Genome genome2, out Genome child1, out
Genome child2)
        {
            int pos = (int)(m_random.NextDouble() * (double)m_length);
            child1 = new Genome(m_length, false);
            child2 = new Genome(m_length, false);
            for(int i = 0 ; i < m_length ; i++)
            {
                if (i < pos)
                {
                    child1.m_genes[i] = m_genes[i];
                    child2.m_genes[i] = genome2.m_genes[i];
                }
                else
                {
                    child1.m_genes[i] = genome2.m_genes[i];
                    child2.m_genes[i] = m_genes[i];
                }
            }
        }


        public void Mutate()
        {
    for (int pos = 0; pos < m_length; pos++)
    {
       if (m_random.NextDouble() < m_mutationRate)
          m_genes[pos] = (m_genes[pos] + m_random.NextDouble()) / 2.0;
    }

        }

        public double[] Genes()
        {
            return m_genes;
        }

        public void Output()
        {
            for (int i = 0 ; i < m_length ; i++)
            {
                System.Console.WriteLine("{0:F4}", m_genes[i]);
            }
            System.Console.Write("\n");
        }

        public void GetValues(ref double[] values)
        {
            for (int i = 0 ; i < m_length ; i++)
                values[i] = m_genes[i];
        }
```

EK – 1 (Devam). Çözüm Programının Kodları

```
        public double[] m_genes;
        private int m_length;
        private double m_fitness;
        static Random m_random = new Random();

        private static double m_mutationRate;

        public double Fitness
        {
                get
                {
                        return m_fitness;
                }
                set
                {
                        m_fitness = value;
                }
        }


        public static double MutationRate
        {
                get
                {
                        return m_mutationRate;
                }
                set
                {
                        m_mutationRate = value;
                }
        }

        public int Length
        {
                get
                {
                        return m_length;
                }
        }

    public int CompareTo(object x)
    {
        if (!(x is Genome))
            return 1;
        else if (((Genome)x).Fitness > ((Genome)this).Fitness)
            return -1;
        else if (((Genome)x).Fitness < ((Genome)this).Fitness)
            return 1;
        else
            return 0;
    }

}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : Can, Mustafa Selim
Uyruğu : T.C.
Doğum tarihi ve yeri: 01.12.1983, Ankara
Medeni hali : Bekar
Telefon : +90 (0) 312 468 53 00 / 4114
e-posta : mselimcan@hotmail.com

Eğitim

Derece		Eğitim Birimi	Mezuniyet tarihi
Lisans	:	Gazi Üniversitesi / Elektrik-Elektronik Müh. Bölümü	2007
Lise	:	Ankara Yıldırım Beyazıt Anadolu Lisesi	2001

İş Deneyimi

Yıl	Yer	Görev
2007-2010	TÜBİTAK	Uzman yardımcısı

Yabancı Dil

İngilizce (Çok iyi)
Japonca (Orta)

Hobiler

Çizgiroman, Karikatür, Animasyon