UÇAK TRANSPONDER SİNYALLERİYLE UÇAK POZİSYONUNUN HİPERBOLİK KONUMLANDIRILMASI

Cengiz PAŞAOĞLU

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCAK 2010
ANKARA

Cengiz PAŞAOĞLU tarafından hazırlanan UÇAK TRANSPONDER SİNYALLERİYLE UÇAK POZİSYONUNUN HİPERBOLİK KONUMLANDIRILMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Nursel AKÇAM …….
Tez Danışmanı, Elektrik-Elektronik Müh.

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

Doç. Dr. Erkan AFACAN …….
Elektrik-Elektronik Müh., Gazi Üniversitesi
Yrd. Doç. Dr. Nursel AKÇAM …….
Elektrik-Elektronik Müh., Gazi Üniversitesi
Yrd. Doç. Dr. Elif URAY AYDIN …….
Elektrik-Elektronik Müh., Atılım Üniversitesi

Tarih: 21/01/2010

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.

Cengiz PAŞAOĞLU

UÇAK TRANSPONDER SİNYALLERİYLE UÇAK POZİSYONUNUN HİPERBOLİK KONUMLANDIRILMASI
(Yüksek Lisans Tezi)

Cengiz PAŞAOĞLU

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Ocak 2010

ÖZET

Havacılık teknolojisi, ülkemiz dahil dünyada hızla gelişmektedir. Hava sahası gözetim sistemleri de havacılık teknolojilerinin en önemli elemanlarıdırlar. Hava sahası gözetim sistemlerinin kullanım amacı, bir Hava Trafik Kontrolörünün gerekli ayırma minimumunu sağlaması için hava/yer aracının pozisyonunu belirlemektir. Başta geleneksel birincil ve ikincil gözetleme radarları olmak üzere kurulumu ve bakımı maliyetli ve zor olan gözetim sistemleri yerine yeni teknolojiler geliştirilmektedir. Multilaterasyon, geliştirilmekte olan hava sahası gözetim sistemlerinden birisidir. Bu sistemler, uçak transponderinden yayılan sinyallerin üç ya da daha fazla anten tarafından alınarak, söz konusu sinyallerin antenlere ulaşım zamanına göre farkının hesaplanmasıyla, uçağın pozisyonunun bulunmasına dayanır. Bu çalışmada söz konusu Multilaterasyon sistemleri ayrıntılı olarak incelenmiş, ulaşım zamanı farkı tekniği kullanılarak bir uçağın pozisyonu iki ve üç boyutlu olarak üretilip çizdirilmiş ve en küçük kareler yöntemi yaklaşımıyla söz konusu uçağın pozisyonu kestirilip simüle edilmiştir.

Bilim Kodu : **905.1.056**
Anahtar Kelimeler : **Hiperbolik konumlandırma, ulaşım zamanı farkı, en küçük kareler yöntemi, uçak transponderi**
Sayfa Adedi : **104**
Tez Yöneticisi : **Yrd.Doç.Dr. Nursel AKÇAM**

FINDING AN AIRCRAFT'S POSITION WITH AIRCRAFT TRANSPONDER SIGNALS USING MULTILATERATION

(M.Sc. Thesis)

Cengiz PAŞAOĞLU

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
January 2010

ABSTRACT

The aviation technology has been developing rapidly in the world as well as in our country. Airspace surveillance systems are the key elements in aviation technology since they have the major role in identifying the aircraft's position to enable air traffic controller to make the necessary separation among the air traffic for flight safety. Today, particularly instead of primary and secondary radars which require more effort to install and much cost to maintain, new technologies, with much lower cost and easy maintenance, are being developed. Multilateration is one of the examples of these new generation surveillance technologies which are being developed. In Multilateration, signals emitted from the aircraft's transponder are received by at least three ground stations and Time Difference of Arrival technique is used to find the location of the aircraft. In this study, the Multilateration systems are examined in detail, the aircraft's location is generated and drawn in two and three dimensions by using time difference of arrival technique, the location of the aircraft is estimated and simulated with the least squares method.

Science Code : 905.1.056
Key Words : Multilateration, TDOA, least squares method, transponder
Page Number: 104
Adviser : Assist. Prof. Dr. Nurscl AKÇAM

TEŞEKKÜR

Bu çalışmada desteğini, bilgisini ve uzmanlığını benden esirgemeyen Sayın Hocam Yrd. Doç. Dr. Nursel AKÇAM'a, beni bugünlere getiren sevgili aileme ve çalışmanın her aşamasında manevi destekleriyle yanımda olan sevgili eşime teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

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

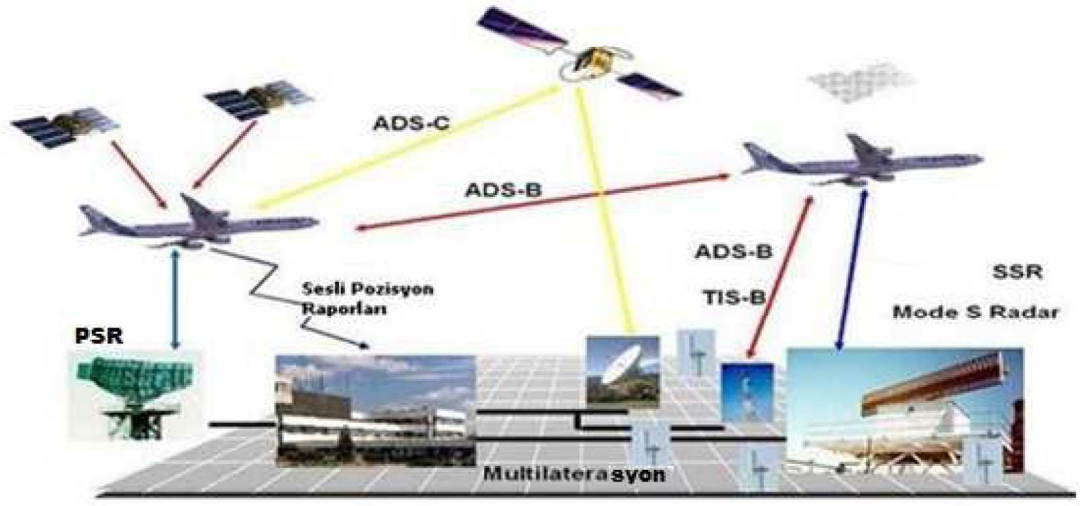
Simgeler	**Açıklama**
A	Tasarım matrisi Nm/sa
c	Işık hızı
d	Uzaklık
ft	Feet
k	Lagranj çarpanı
L	Ölçüm değeri
MHz	Megahertz
Nm	Deniz mili
T	Ulaşım zamanı
τ	Ulaşım zamanı farkı

Kısaltmalar	**Açıklama**
AA	Duyurulan Adres
AC	İrtifa Kodu
ACAS	Havaaracı Çarpışma Önleme Sistemi
ADS-B	Otomatik Bağımlı Gözetim Yayımı
ADS-C	Otomatik Bağımlı Gözetim Kontratı
AP	Adres Parite
ASMGCS	Yer Hareketleri Rehberlik ve Kontrol Sistemi
CA	Kapasite
CDM	İşbirlikçi Karar Yapımı
CNS	Kominikasyon Seyrüsefer Gözetim
DF	Aşağı Bağlam Formatı

DME	Uzaklık Ölçüm Ekipmanı
DR	Aşağı Bağlam İstemi
EUROCONTROL	Avrupa Hava Seyrüseferi Emniyeti Teşkilatı
ES	Uzatılmış Periyodik Veri Yayımı
FS	Uçuş Statüsü
GPS	Küresel Konumlandırma Sistemi
HTK	Hava Trafik Kontrolörü
IATA	Uluslararası Hava Taşımacılık Örgütü
ICAO	Uluslararası Sivil Havacılık Organizasyonu
ID	Tanıtım
IFF	Dost ya da Düşman Tanıtımı
LAM	Yerel Alan Multilaterasyon
MB	B Mesajı Alanı
MDA	En Düşük Karar İrtifası
MLAT	MultilaterasyonHiperbolikKonumlandırma
Mode S	Seçici Mod
PI	Parite Bilgisi
PSR	Birincil Gözetim Radarı
RA	Radyo Altimetre
RVSM	Azaltılmış Dikey Ayırma Minimumu
SMR	Yüzey Hareket Radarı
SSR	İkincil Gözetim Radarı
TCAS	Trafik Çarpışma Önleme Sistemi
TDOA	Ulaşım Zamanı Farkı
TIS-B	Trafik Bilgisi Gözetim Yayımı
TMA	Terminal Alan
TOA	Ulaşım Zamanı
UHF	Ultra Yüksek Frekans
VHF	Çok Yüksek Frekans
WAM	Geniş Alan Multilaterasyon

1. GİRİŞ

Hava sahası gözetim sistemleri, dünyada özellikle sivil uçuşların her geçen gün giderek artma eğilimi göstermesiyle büyük önem kazanmıştır. Gözetim unsuru ilk olarak manuel kontrolle başlamış (sesli pozisyon raporları), radarların geliştirilmeye başlamasıyla ivme kazanmış, günümüzde de MLAT (Multilaterasyon; Multilateration) ve ADS-B (Otomatik Bağımlı Gözetim Yayımı; Automatic Dependent Surveillance Broadcast) gibi daha ucuz ve daha doğru pozisyon bilgisi elde edilebilen sistemler üzerine yapılan çalışmalarla devam etmektedir. Resim 1.1'de günümüzde kullanılan ve yeni geliştirilmekte olan hava sahası gözetim sistemleri bir arada verilmiştir.



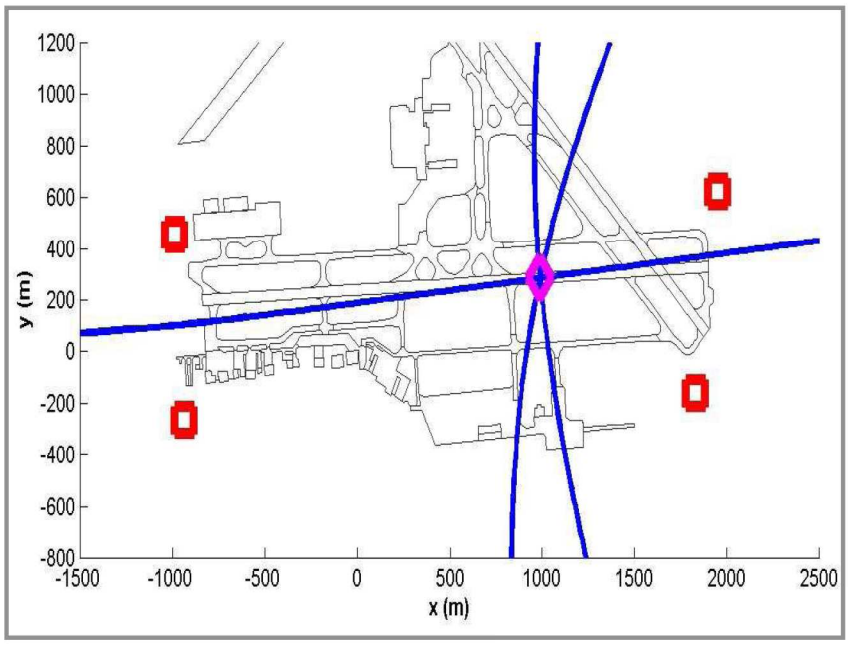
Resim 1.1. Havasahası gözetim sistemleri

20. yüzyılın başlarında radyo dalgalarının özellikleri ve uygulama prensipleri konusunda çalışmalar yapılmaya başlandı. Araştırmalar esnasında radyo dalgalarının bir kısmının objelere çarparak geri döndüğü fark edildi. İkinci dünya savaşı sonrasında yapılan çalışmalar sonucunda ilk sivil radar meydan civarındaki iniş ve kalkışlara hizmet vermek amacıyla kullanılmaya başlandı. Kullanılan bu birincil gözetleme radarı (PSR; Primary Surveillance Radar), yer istasyonundaki kendi ekseni etrafında 360 derecelik dönüş yapabilen bir anten yardımı ile yüksek hızda havaya

elektromanyetik radyo dalgaları gönderen bir sistemdi. [2]. Bu radyo dalgaları karşılaştıkları herhangi bir hedeften yansıyarak antene geri döner ve antendeki özel bir alıcı tarafından işlenirler. Işık hızında hareket eden dalgaların gönderilme zamanı ve bir hedeften yansıyarak geri dönmesi zamanına göre, hedefin yeri ve mesafesi radar tarafından tespit edilerek radar ekranı üzerinde gösterilir.

Daha sonra, manuel kontrol genel olarak devam etmesine rağmen 20–50 NM (Deniz Mili; Nautical Mile) gözetleme kapasitesine ulaşan radarlar, yaklaşma kontrol hizmetinde kullanılmaya başlandı. Birçok noktaya PSR istasyonları kurulmuş, PSR'lerin saha kontrol hizmetinde de kullanılmasına başlanmıştı. Ancak hava sahasında birbirine benzeyen birçok hedef karışıklık yaratmaya başlayınca ve hedeflerin takibi zorlaşınca; uçakları daha kolay tanımlama ve uçuş seviyelerini görebilme imkanları geliştirildi. Böylece PSR'ye ek olarak uçak transponderleri ve SSR (İkincil Gözetleme Radarı; Secondary Surveillance Radar)'ler geliştirilmeye başlandı [2-4]. Uçaklarda bulunan transponderler SSR'lerin 1030 MHz'de gönderdikleri sorgu sinyallerine 1090MHz'de cevap sinyalleri üreterek gönderirler ve bu cevap sinyallerinde bulunan bir tanımlama kodu yardımıyla uçaklar ayrı ayrı tanımlanarak birbirinden ayırt edilebilirler.

İlk MLAT fikri ise 1990'lı yılların ortalarında ortaya çıkmıştır. Transponderlerden 1090 MHz'de gelen sinyaller kullanılarak, belirli bir sahaya yerleştirilecek antenler ile uçak pozisyonunun SSR'lere nazaran çok daha doğru bir şekilde kestirilebileceği fark edilmiştir [9-12]. Fakat havacılıkta en önemli unsur güvenlik olduğu için günümüze kadar MLAT ile ilgili çalışmalar devam etmiştir. Ayrıca maliyet açısından çok daha kazançlı olduğu için, radarların yerini yavaş yavaş bu ve benzeri sistemlerin alması beklenmektedir. MLAT sistemleri işbirlikçi (uçaklarda transpondere ihtiyaç olduğu için) bağımsız (pozisyon sistem tarafından hesaplanarak bulunduğu için) gözetim sistemleridirler. Şekil 1.1'de örnek bir MLAT sistemi gösterilmektedir.



Şekil 1.1. MLAT sistem örneği

MLAT sistemler, uçak transponderinden yayılan sinyallerin en az 3 alıcı anten tarafından yakalanıp TDOA (Ulaşım Zamanı Farkı; Time Difference of Arriaval) yöntemi kullanılarak uçak pozisyonunun bulunması temeliyle çalışan sistemlerdir.

Bucher ve Mısra [5], üç boyutlu hiperbolik pozisyonlama sistem algoritması için VHDL (Very high speed integrated circuit Hardware Description Language) modeli geliştirmişlerdir. Söz konusu çalışmada 4 tane sabit alıcı istasyonu ile bir mobil telefonun pozisyonunu elde etmişlerdir.

Brown, Hardiman ve Carter [6], MLAT pozisyonlama için Lineer olmayan kestirim teknikleri konusunda çalışmalar yapmışlardır. Anılan çalışmalarda insansız sistemlerin kendi pozisyonlarını hesaplamaları ve GPS (Küresel Konumlandırma Sistemi; Global Posistioning System) benzeri hizmet verebilmeleri için geliştirdikleri yöntem anlatılmaktadır.

Trofimova [7], MLAT konusunda hata araştırmaları ve sınıflandırmaları yapmıştır. Hata kestirimi çalışmalarını gerçekleştirmiştir.

Bu tez çalışmasının ikinci bölümünde uçak transponderleri hakkında bilgiler verilmiştir. Halihazırda kullanılmakta olan transponderler ve fonksiyonları anlatılmış, transponderlerden yayılan sinyaller incelenmiştir.

Üçüncü bölümde MLAT sistemlerle ilgili genel bilgiler verilmiş, kullanım alanları, sistem yapıları ve çeşitleri anlatılmıştır.

Dördüncü bölümde TDOA yöntemi anlatılmış, pozisyon yaklaşımları için kullanılan en kücük kareler yönteminden bahsedilmiş, simülasyonun oluşturulacağı formüller çıkarılmış ve incelenmiştir.

Beşinci bölümde iki ve üç boyutlu simülasyon çalışması için oluşturulan fonksiyon ve programlar detaylarıyla anlatılmıştır.

Altıncı bölümde elde edilen simülasyonların çalıştırılması sonucu oluşan çıktılar örneklerle verilmiş, hata oranlarına göre değerlendirmeler yapılmıştır.

Sonuç bölümünde ise tasarlanan simülasyonlardan elde edilen sonuçlar açıklanmış ve bu sonuçlara göre öneriler sunulmuştur.

2. UÇAK TRANSPONDERİ

Uçak transponderleri, ikinci dünya savaşı sırasında İngiliz ve Amerikalılar tarafından radar üzerinde, askeri amaçlı dost ve düşman uçaklarını ayırt edip tanımlamak için geliştirilmişlerdir. Kavram olarak da IFF (Dost ya da Düşman Tanıtımı; Identification Friend or Foe) kullanılmış ve 1950'lerde sivil hava trafik kontrolü ile genel havacılık ve ticari havacılık için hava trafik hizmetleri sağlamak üzere ikincil gözetim radarı sistemleri kullanılarak uyarlanmışlardır [2].

Transponder, "transmitter-responder" terimlerinin kısaltılmasıyla oluşturulmuş bir kelimedir. Radyo frekansında bir sorgulama aldığında cevap yayımlayan elektronik aletlerdir. Havacılıkta transponderler temel olarak uçakları SSR'ler vasıtasıyla tanımlamak ve diğer bir uçakla çarpışmaları önlemek için kullanılırlar. Resim 2.1'de uçaklarda kullanılmakta olan eski ve yeni tip transponderler görülmektedir.

(a)

(b)

Resim 2.1. Örnek transponderler

a) Eski (analog)

b) Yeni (mod S)

2.1. Transponder Modları

Uçak transponderi için birçok farklı cevap modları vardır. Bu modlar bir SSR anteninden alınan sorgu sinyaline karşı iletilen cevap sinyalleridirler:

Mod 1: 5 bitlik 2 haneli görev kodudur (sadece askeri amaçlı).
Mod 2: 4 haneli oktal askeri birlik kodudur (sadece askeri amaçlı).
Mod 3/A: 4 haneli sekizlik uçak tanımlama kodudur. Hava trafik kontrolörü tarafından uçağa atanır (hem sivil hem askeri amaçlı kullanılmaktadır).
Mod C: Uçağın basınç irtifası için 4 haneli oktal koddur.
Mod S: Seçici sorgulama için çoklu bilgi formatı sağlayan koddur. Her bir uçak için sabit bir 24 bitlik adres atanmıştır.

Transponder'den gönderilen Mod A-B bilgisi yayının modunu, Mod C de uçağın irtifasını (uçuş seviyesi türünden) verir. Transponder'ler Mod A ile birlikte 4 basamaklı (0-7 arasındaki rakamlardan oluşan) bir kod bilgisi gönderirler. Buna SSR kodu denilir. İki çeşit SSR kodu vardır:

2.1.1. Blok kodlar

Hava aracının gönderdiği 4 basamaklı kodların ilk iki basamağı kullanılır (4567 için 45, 2731 için 27 v.b.). Toplam 64 adet blok kod vardır. Bu kodlar eski radar sistemlerinde kullanılırlar. Her sektör için farklı bir kod vardır ve ilgili sektörün sorumlu hava trafik kontrolörü, kontrolü altındaki bütün hava araçlarına o sektör için belirlenmiş kodu bağlatır, böylece diğer sektörlerdeki hava araçlarıyla kendi sorumluluğundaki hava araçlarını ayırmış olur.

2.1.2. Ayrık kodlar

Hava aracının gönderdiği 4 basamaklı kodların tamamı kullanılır (4567, 2731 v.b.). Toplam olarak 4096 adet ayrık kod vardır. Bütün hava araçlarına farklı kodlar bağlatılır. Kontrolör, belli bir kodu bağlayan uçağın çağrı adını radar sistemine girerek trafiklerin çağrı adıyla takip edilmesini sağlar.

Özel kodlar:

- A7700 – Acil durumu olan trafikler
- A7600 – Radyo iletişim kaybı
- A7500 – Uçak kaçırma

Türkiye radar sistemlerinde ayrık kodlar kullanılmakla birlikte, blok kodlar da kullanılabilir [2-3].

2.1.3. Mod S

Günümüzde artan hava trafiği nedeniyle mevcut 4096 adet SSR ayrık kod yetersiz kalmaktadır. Bu amaçla geliştirilen Mod S fonksiyonu sayesinde her hava aracı için 24 bitlik sabit birer ICAO (Uluslararası Sivil Havacılık Organizasyonu; International Civil Aviation Organization) adresi olacaktır. Tam olarak 16 777 214 hava aracı için sabit kod tahsisi yapılabilecektir. Uçuştan önce pilot cihazlarına uçuş planındaki çağrı adını girer ve bütün Mod S sistemlerinde çağrı adı ile görüntülenir. Bu sayede hava aracı ile ilgili her türlü bilgiye de ulaşabilmek mümkün olur [2].

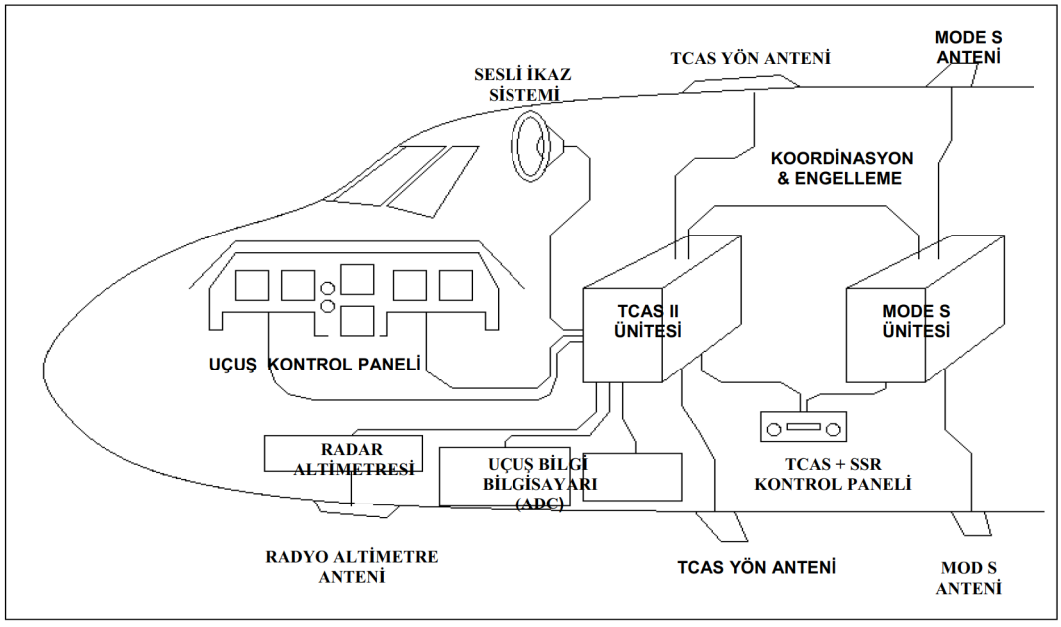
2.2. Transponder Sinyalleri

Uçaklar günümüzde CNS (İletişim Navigasyon Gözetim; Communication Navigation Surveillance) amaçlarını gerçekleştirmek için birçok anten ile donatılmışlardır.

Şekil 2.1'den görüleceği üzere uçak üzerindeki karmaşık yapı yüzünden uçak şirketleri genellikle uçaklara herhangibir ekstra cihaz/alet eklenmesini istemezler. Zira uçaklara eklenecek yeni cihazların diğer cihazlarla girişim yapmaması, kablolama vs. çalışmalarının çok dikkatli yapılması gerekir. Bunun için de çok maliyetli testler ve sertifikasyon işlemleri gerekir. Dolayısıyla havacılık sanayinde geliştirilen yeni sistemlerin uçaklara herhangi bir yük getirmemesine özellikle dikkat edilmektedir. MLAT siteminin en önemli avantajlarından birisi de uçak üzerinde halihazırda zaten var olan sistemlerin ve sinyallerin kullanılmasıdır.

Havacılık sanayiinde uçak tarafından gönderilen sinyallerin kullanılabilirliği aşağıdakilere bağlıdır:

- Uçak tanıtımı/kimliği
- Sinyal geçerliliği/uçaktaki kullanım oranı
- Güncellenme derecesi
- Algılanabilme performansları



Şekil 2.1. Uçak iç yapısı ve elemanları

2.2.1. İletişim sinyalleri

İletişim sinyalleri uçakların hava trafik kontrol merkezleri ve/veya şirket yetkilileri ile haberleşmek için kullandıkları sinyallerdir. VHF/UHF üzerinden sesli iletişim yolu ile haberleşme yapılır. Herhangi bir uçak tanıtımı/kimliği yoktur, sinyal geçerliliği sınırlı ve düzensizdir, sadece pilot ya da kontrolörün elindeki butona basmasıyla haberleşme gerçekleştirilebilir.

2.2.2. Navigasyon sinyalleri

Adından da anlaşılacağı üzere uçağın navigasyonunu yapması için kullanılan sinyallerdir. Bu sinyaller uçak kimlik bilgilerini içermezler. MLAT için uçak açısından bakıldığında sadece aktif navigasyon teknikleri ilgili olabilir. Bunlar:

- Radyo Altimetre (RA; Radio Altimeter)
- Uzaklık Ölçüm Ekipmanı (DME; Distance Measurement Equipment)'dır.

Radyo altimetre navigasyon sinyalleri

RA radyo dalgaları vasıtasıyla uçağın gerçek yüksekliğinin bulunması amacına hizmet eder. Uçaktan düşey olarak gönderilen dalgaların yerden yansıyıp dönme süresinin ölçülmesi ve buna bağlı olarak da uçağın irtifasının bulunması prensibine dayanır [1]. Yüksek irtifa ve düşük irtifa radyo altimetresi olmak üzere iki çeşidi vardır. MLAT için kullanışlı değildir.

DME navigasyon sinyalleri

Pilota yer istasyonu ile uçak arasındaki uzaklığı veren ve UHF bandında yayın yapan bir sistemdir. DME prensibi, SSR'ye benzmektedir. Hem uçakta hem de yer istasyonunda alıcı ve verici anten vardır. Uçağın gönderdiği sorgu sinyalleri yer istasyonunda değerlendirilir ve farklı bir frekansta cevap sinyali olarak uçağa gönderilir. Yer istasyonunda sorgu sinyalinin değerlendirilmesi 50 ms'de

gerçekleştirilir [1]. Sorgu ve cevap sinyali arasında 63 MHz'lik frekans farkı vardır. Sistemin çalışma prensibi, sinyalin gidiş-dönüş süresinin ölçülmesine dayanır. İletişimde darbe modülasyonu kullanılır. 250 NM'e kadar en-route (uçağın belli bir irtifanın üstündeki yol aşaması) için tasarlanmıştır ve uçakların standart ekipmanı olduğu için kullanım oranı çok yüksektir. Bu sinyallerin güncellenme derecesi 15 ile 1502 Hz arasındadır [1-2]. MLAT açısından bakıldığında ise uçak kimlik olasılığının az olması kullanışını sınırlamaktadır. Fakat SSR transponder kapatıldığında back up (yedek) olarak kullanım olasılığı vardır.

2006 yılında DME navigasyon sinyalleri kullanılarak MLAT çalışmaları yapılmıştır ve elde edilen sonuçlara göre DME navigasyon sinyallerinin kullanılabilirliği kanıtlanmıştır. Bu çalışmalarda uçaklar ayırt edilebilmiş, uçak pozisyonu hesaplanabilmiş ve uygulanabilirlik gösterilmiştir [1].

2.2.3. Gözetim sinyalleri

Gözetim sinyalleri uçakların gözetim ve tanımlamasını yapabilmek için kullanılan sinyallerdir. Havacılıktaki kullanımı açısından bu sinyalleri PSR gözetim sinyalleri ve 1090MHz gözetim sinyalleri diye ikiye ayırabiliriz.

PSR gözetim sinyalleri

Bir birincil radar, hedefe yüksek frekanslı sinyaller gönderir. Hedeften yansıtılan sinyaller radar tarafından alınır ve değerlendirilir. Birincil radarın en önemli özelliği, pasif yansımalarla çalışmasıdır. Yansıyan sinyallerin kaynağı radar ünitesi tarafından gönderilen darbelerdir. Yol kontrol ve yaklaşma safhalarında bu sinyaller yüksek oranda bulunurlar. ICAO kurallarına göre yaklaşma safhasında mutlaka PSR radar bulunmalıdır. Ayrıca ülkemizde de radarlı yaklaşma hizmeti verilen meydanlarımızda PSR ve SSR radarlar on-mounted (üst üste) olarak birlikte çalışmaktadırlar [1-3,13].

PSR gözetim sinyallerinde, uçak tanıtımı/kimlik bilgisi yoktur. Bu radarlar değişik frekans bantları kullandıkları için evrensel bir alıcı yapmak da oldukça zordur ve

yeryüzünün yarattığı gürültülere (clutter) karşı da hassastırlar [2]. Dolayısıyla MLAT için dezavantajları çok daha fazladır.

1090 MHz gözetim sinyalleri

Söz konusu 1090 MHz gözetim sinyalleri

- Mod A/C SSR cevapları,
- Mod S cevapları,
- Mod S Kısa periyodik veri yayımı (squitter),
- Mod S Uzun/Uzatılmış periyodik veri yayımı,

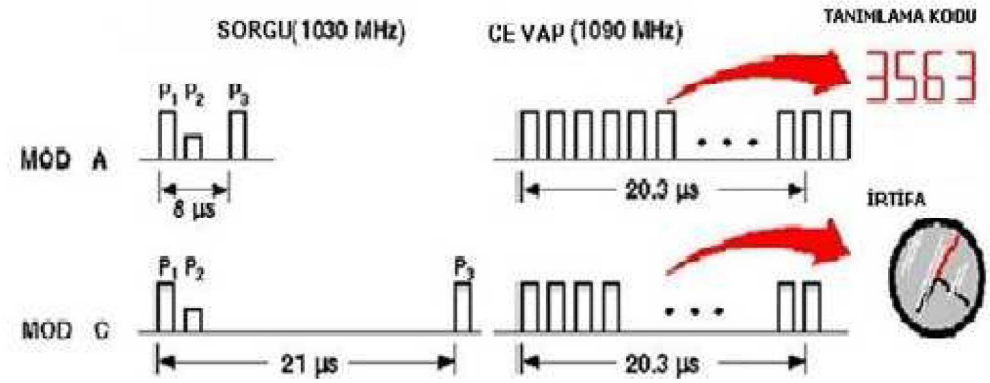
sinyallerinden oluşur. Bir alıcı bu dört sinyal türünü de çözebilmektedir [1-3].

Mod A/C SSR cevapları

Aslında klasik SSR cevapları 1090 MHz'de üretilen ve temel bilgileri taşıyan cevaplardır. SSR'nin kullanımındaki kabul, tüm "dost" uçakların bir transponder taşımasıdır. Sorgulayıcı, hayati önem taşıyan iki soru sorar: "Neredesiniz ?" ve "Kimsiniz ?" [1]. Birincisi radar tekniğinin kullanımıyla yanıtlanabilir. Hem sorgulayıcı hem de transponder, darbe (pulse) iletimini kullanır. Sorgulamadan, alındı cevabına kadar geçen süreden aradaki mesafe hesaplanabilir (birincil radarda olduğu gibi). "Kimsiniz ?" sorusunun cevabı ise, pozisyon kodlu darbe gruplarının formu dikkate alınarak yanıtlanır.

Sorgulayıcı, 1030 MHz frekansında bir darbe çifti yayınlar. Her bir darbe aynı şekil, aynı genlik ve aynı süreededir. Bunların zaman aralıkları (spacing), daha önceden tanımlanmış standartlardadır. Böylece, darbe pozisyon kodlaması, çeşitli soruları veya "sorgulama modlarını" ayırt eder [1-2].

Bir kontrol darbesi olan P2, sorgulama darbe çifti P1 ve P3'e eşlik eder. Alma menzilindeki transponder'ler darbe çiftini tespit eder, zaman aralıklarını algılar ve sorgulamanın genel durumuna uygun cevap oluştururlar. Eğer, bir transponder, 8 µs'lik aralığı algılarsa, otomatik cevap, aracın kimliğini taşıyan bir kod olacaktır (Mod A). Hava trafik kontrol sistemlerinde, uçak yüksekliğinin büyük önemi vardır ve 20.3 µs aralığa sahip sorgulama darbe çiftine, kodlu formda, uçak yüksekliği otomatik olarak verilir (Mod C) [2]. Şekil 2.2'de mod A/C cevap sinyalleri açık bir şekilde görülmektedir.



Şekil 2.2. Mod A/C cevap sinyalleri

Mod A/C gözetim sinyalleri sadece uçak havada olduğunda ve yerde de klasik bir SSR radarı olduğunda gönderilirler. MLAT sistemlerde kullanılabilirliği açısından baktığımızda en önemli avantajları, çok yüksek oranda kullanılıyor olmaları ve mod A ile uçak tanıtımının elde edilebilir olmasıdır. Fakat mod A kodunun her zaman sabit olmaması, mod A ve mod C kodlarının ayırt edilebilmesinin bazen gerçekten zor olması, yüksek yoğunlukta trafik olan alanlarda girişimlerin olması ve güncelleme derecesinin radar konfigürasyonuna bağlı olması da yadsınamayacak dezavantajlarıdır.

Mod S cevapları

Yoğun hava trafiğinin olduğu yerlerde mod A kod sıkıntısı, transponderlerin çok fazla sorgulanması sonucu kilitlenmesi, uçak hakkında ilave bilgi ihtiyacı,

elektromanyetik kirlilik gibi sebeplerden dolayı mod S radarın geliştirilmesine ihtiyaç duyulmuştur [4].

SSR radarları, klasik olarak önce uçaktan mod A sorgulaması yapar, dinlemeye geçer ve daha sonra yükseklik bilgisi olan mod C sorgulaması yaparak dinlemeye geçer. Mod S radarları ise klasik radar çalışma prensibine de uyumlu olacak şekilde önce tüm normal transponderli uçakları sorgular ve bu arada mod S radarı olduğundan ekstra bir P4 darbesi gönderir. Bu P4 darbesini algılayan mod S transponderler cevap vermezler. Bu tüm sorgulamanın ikinci aşamasında ise P4 darbesi uzun tutularak mod S transponderlerin de cevap vereceği tüm sorgulama yapılır ve bütün mod S ve klasik SSR transponderler mod A ve mod C cevaplarını verirler. Bu aşamada klasik transponderler uçağın mod A ve mod C bilgilerini verir. Mod S transponderler ise bu bilgilerle birlikte uçağın kimlik bilgilerini de verir [1-2]. Bu bilgiyi alan mod S radarları o azimutta o uçağın olduğunu hafızasına alır ve sadece mod S transponderleri için olan ikinci sorgulamasında, bütün sorgulamada kimlik bilgilerini elde ettiği uçakları sadece seçici olarak sorgular ve aynı zamanda kendine ait olan tanıtım kodunu da verir. Böylece her mod S transponderi kendisini hangi mod S radarı sorguluyorsa sadece ona cevap verir [1].

Şekil 2.3'te mod S cevap sinyalleri gösterilmiştir. Bunlar kısa cevaplar ve uzun cevaplar olmak üzere ikiye ayrılırlar. İlk beş bitlik alan format sayısını göstermektedir. Daha sonra sırasıyla FS (Uçuş Statüsü/ Uçağın yerde mi, havada mı olduğunu gösteren alan; Flight Statue), DR (Aşağı Bağlantı İstemi; Downlink Request), UM (Kullanılabilirlik Mesajı; Utility Message), AC (İrtifa Kodu; Altitude Code), ID (Uçak Tanıtımı; Identification), MB (B mesajı alanı; B Message Field) ve AP (Adres Parite; Address Parity) alanları vardır. Bu alanlardan MLAT açısından daha önemli alanlar format sayısı alanı, FS, AC/ID, MB ve AP alanlarıdır.

Kısa mod S cevapları için iki özel çeşit vardır. Bunlar DF04 (Aşağı Bağlantı; Downlink Format) ve DF05 formatlarıdır. DF04 AC içeren gözetim cevabı, DF05 ise ID içeren gözetim cevabıdır.

Uzun mod S cevapları da DF20 ve DF21 olmak üzere iki çeşittir. DF20 AC içeren ve uçağın gerçek zamanlı bilgisini taşıyan cevaplar, DF21 ise ID içeren ve uçağın gerçek zamanlı bilgisini taşıyan cevaplardır. Bu gerçek zamanlı bilgiler, uçağın çağrı adı, uçuş başı, hızı, seçilmiş dikey hareket gibi bilgileri içermektedirler.

Mod S gözetim sinyallerinin kullanılabilirliği her geçen gün artmaktadır. 24 bitlik adres üzerinden her uçak tek ve kesin bir şekilde tanımlanabilmektedir ve 25 ft'lik doğruluğu olan irtifa ölçümleri yapılabilmektedir. Bunun yanında yüksek yoğunlukta trafik olan yerlerde girişimler olabilmekte ve güncelleme derecesi de radar konfigürasyonuna bağlı olmaktadır [1,4].

5	3	5	6	13	24
04 05	FS	DR	UM	AC/ID	AP

(a)

5	3	5	6	13	56	24
20 21	FS	DR	UM	AC/ID	MB	AP

(b)

Şekil 2.3. Mod S cevap mesajları

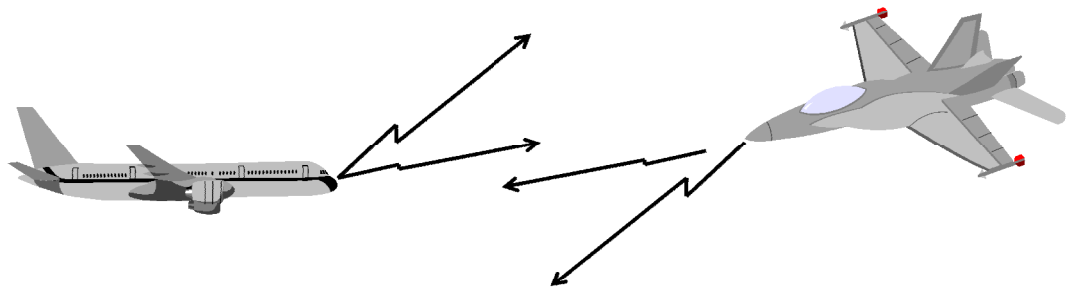
a) Kısa mod (DF04/05)

b) Uzun mod (DF20/21)

Mod S kısa (short) periyodik veri yayımı (DF11)

Squiter terimi talep edilmemiş cevaplar demektir. Yani bir uçağın transponderinin herhangi bir sorgulama olmadan/kendiliğinden belirli aralıklarla gönderdikleri

bilgilerdir. Bu terim ilk önce uçakların havada çarpışmasını önlemek için geliştirilen TCAS/ACAS (Hava Aracı Çarpışma Engelleme Sistemi; Traffic/Airborne Collision Avoidance System) sistemleri ile kullanılmıştır [2]. Uçaklar Mod S transponderleri vasıtasıyla saniyede bir kere belli bir mesafeye bilgilerini yaymaktadırlar. Böylece uçaklar birbirlerine olması gerekenden fazla yaklaştığı zaman bu sistem sayesinde uyarı alarak kaçınma yapabilmektedirler. Şekil 2.4'te mod S kısa periyodik veri yayımı mesaj örneği (DF11) gösterilmiştir.



5	3	24	24
11	CA	AA	PI

Şekil 2.4. Mod S kısa periyodik veri yayım mesajı

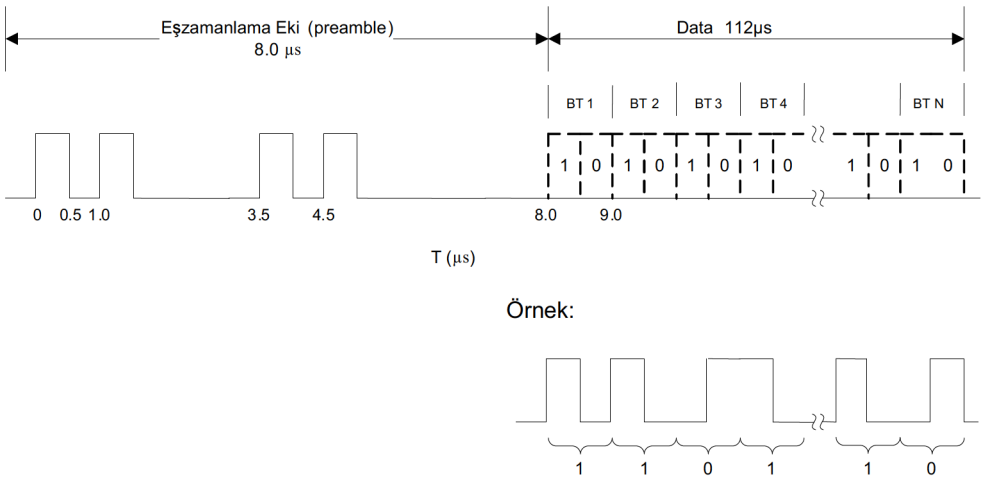
Burada ilk beş bit gene format numarasıdır. Sonrasında gelen 3 bitlik CA (kapasite/ transponder seviyesi ya da herhangi aktif bir alarm veya tanıtma varsa onu içerir), AA (İlan Edilen 24 bitlik adres bilgisi; Address Announced), PI (Eşlik Bilgisi/Eşlik Kontrolü; Parity Information) demektir. Bu sinyaller 8 bitlik eşzamanlama ekiyle birlikte 64 µs sürmektedirler. Uçaklarda TCAS/ACAS taşıma zorunluluğu olduğu için kullanılabilirlikleri çok yüksektir [2]. 24 bitlik adresten dolayı uçak tanımı tektir, spontane gönderilen sinyaller oldukları için sorgulayıcıya gerek yoktur ve güncelleme derecesi 1 Hz'dir, dolayısıyla MLAT sistemler için çok avantajlıdırlar.

Mod S uzatılmış (extended) periyodik veri yayımı (DF17)

Mod S uzatılmış periyodik veri yayımında (ES), kısa periyodik veri yayımına göre en önemli fark Şekil 2.6.'dan da görüleceği gibi ekstradan 56 bitlik uçak datasını

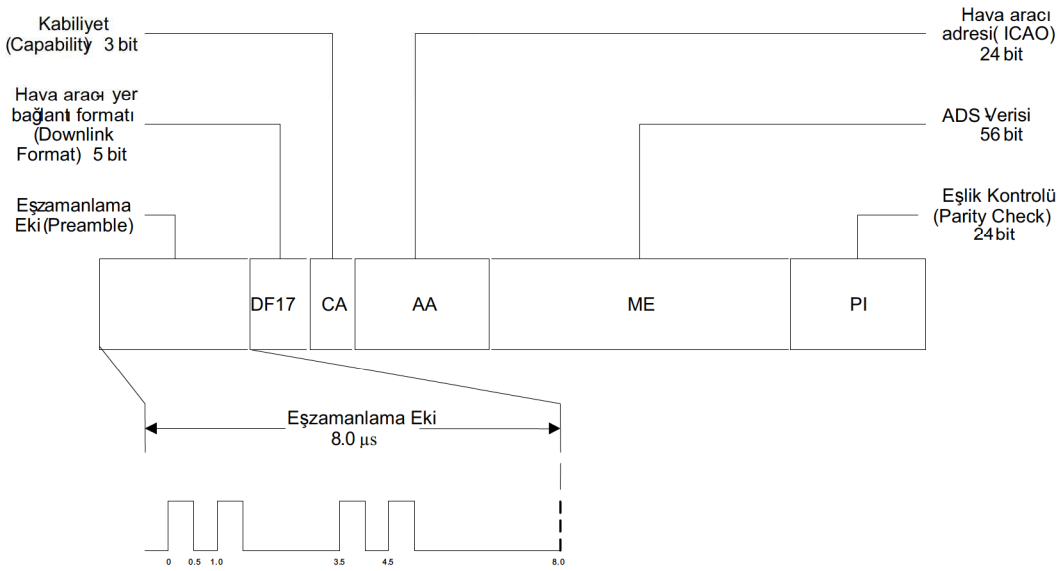
içermesidir. ADS-B sistemlerini oluşturan uçaktaki en önemli elemandır. ES olmazsa uçaktan ADS-B bilgisi alınamaz ve gözetim yapılamaz.

ES mesajları darbe konum modülasyonu (PPM; Pulse Position Modulation) yöntemi ile iletilirler. PPM yönteminde darbenin genlik ve genişliği sabit tutulur. Genlik ve süre sabit olduğu için sabit iletici gücü gerektiren sistemlerde bu yöntem kullanılır. Bu yöntemin bir dezavantajı gönderici-alıcı (transmitter-receiver) arasında senkronizasyon gerektirmesidir. Şekil 2.5'te PPM gösterimi mevcuttur. Şekilde ve örnekte görüldüğü gibi darbe, bit periyodunun ilk yarısında (1) ya da ikinci yarısında iletilir (0) [3].



Şekil 2.5. ES mesajlarının PPM ile iletimi

ES mesajları senkronizasyonunu sağlamak için 8 µs'lik eşzamanlama eki kullanır. Uçak verisi (genelde ADS verisi olarak geçer) 112 µs'lik sürede iletilir ve toplam 120 µs'lik bir süre gerekir. Veri uzunluğu 112 bit'tir ve Şekil 2.6'da mesajın açık hali gösterilmektedir. ADS mesajlarının alınması/algılanması 1090 MHz dalga şekli eşzamanlama ekinin tanınması, eşlik bit kontrolleri, hata kontrolü ve gerekirse hata düzeltmesini kapsar.



Şekil 2.6. Eşzamanlama eki ile birlikte ADS-B mesajı

ES sinyalleri de kısa periyodik veri yayımı sinyalleri gibi MLAT kullanımı için gerek sorgulamaya ihtiyaç olmaması, gerekse uçak tanımının 24 bitlik adres üzerinden kesin bir şekilde yapılabilmesi açısından çok avantajlıdırlar. Ayrıca bu sinyallerin güncelleme derecesi de 6.2 Hz'e kadar çıkmaktadır.

Yukarıda bahsedilen sinyallerden başka bölgesel olarak kullanılan sinyaller de vardır. Bunlardan ES benzeri kullanılanlardan en önemlileri, 108-137 MHz frekanslı İngiltere Kuzey Denizi petrol ve gaz platformu uygulamalarındaki VDL-4 (VHF Datalink Mode 4) ve daha çok ABD'de tercih edilen 978 MHz'te kullanılan UAT (Universal Access Transceiver)'dir [1].

Çizelge 2.1'de 1090 MHz'de kullanılan sinyallerin temel karakteristik özellikleri, Çizelge 2.2'de de MLAT sistemler için kullanılabilecek sinyallerin karakteristik özellikleri özet halinde verilmiştir.

Çizelge 2.1. 1090 MHz sinyallerin temel karakteristik özellikleri

Sinyal Tipi	Ekipman mevcudiyeti	Tetikleme kaynağı	Derece /s	Kaynak tanımlama	Elde Edilen Bilgi
Mod A/C cevapları	Transponderlerin hepsi	SSR radarları TCAS (sadece mod A/C uçakları)	0 - 500	Sınırlı (A/C kodu)	Mod A Mod C
Mod S	97% (yerel)	Mod S radar TCAS	0 - 60	24 bit Adres	Mode A İrtifa Uçak tanımı/kimliği ++
Kısa periyodik veri yayımı	Tüm Mod S uçakları	-	1	24 bit Adres	24 bit Adres
Uzatılmış periyodik veri yayımı	70% (yerel)	-	4 - 6.2	24 bit Adres	Mode A İrtifa Uçak tanımı/kimliği + (pozisyon …)

Çizelge 2.2. MLAT için kullanılabilecek sinyallerin temel karakteristikleri

	TANIMLAMA	EKİPMAN MEVCUDİYETİ	SİNYAL ÖZELLİKLERİ
PSR	Hayır	Yüksek	Zayıf
SSR	Mod 3/A	Çok yüksek	İyi
SSR MOD S	Mod 3/A, 24 bit adres	Yüksek	İyi
MOD S Kısa Periyodik Veri Yayımı	24 bit adres	Yüksek	İyi
MOD S Uzatılmış Periyodik Veri Yayımı	24 bit adres	Yüksek	İyi
VDL MOD 4	24 bit adres	Sadece bölgesel	Zayıf
UAT	24 bit adres	Sadece bölgesel	Ortalama
Radyo Altimetre	Hayır	Yüksek	Zayıf
DME	Hayır	Yüksek	Ortalama
VHF DF	Hayır	Yüksek	Zayıf
ACARS	Hayır	Yüksek	Zayıf
VDL MOD 2	Hayır	Artmakta	Zayıf
Uçak Meteroloji Radarı	Hayır	Yüksek	Zayıf

3. MLAT SİSTEMLER VE UYGULAMA ALANLARI

MLAT on yıllardır kullanılan, kanıtlanmış bir teknolojidir. İlk başlarda askeri amaçlı olarak bir uçağın yerini, (birçoğu görünmek istemez) TDOA tekniğini kullanarak tespit etmek için geliştirilmiştir [1-2]. MLAT sistemlerde bir havaalanını, onun terminal sahasını ya da daha geniş alanlı hava sahalarını kapsamak için birçok yer istasyonu kullanılır.

MLAT yer istasyonları bütün transponderi olan uçaklardan gelen (radar ve ADS-B aviyonikleri dahil) cevap sinyallerini alarak TDOA tekniğine dayalı olarak uçak pozisyonunu hesaplarlar.

Aslında sistem terslenmiş GPS sistemi olarak da düşünülebilir. Uçak uydu olarak düşünülürse antenler de hedef olabilirler [10].

Bir MLAT sistemi şu elemanlardan oluşur:

- Sorgu mesajlarını içeren iletici altsistem, iletim fonksiyonu,
- Hedeften iletilen sinyalleri yakalayan alıcı anten dizisi,
- MLAT izlerini hesaplayıp üreten merkezi işlemci [8,10].

Eğer sistem elemanları içerisinde uçak trasnponderlerini aktif hale getirmek için 1030 MHz'de sorgulayıcı bir alt sistem kullanılıyorsa sistem aktif MLAT, kullanılmıyorsa pasif MLAT olarak geçer.

Dünyada MLAT sistemleri çeşitli amaçlarla kullanılırlar. Bunlar:

- Havaalanı yüzeyi,
- Terminal alan,
- Geniş alan (Wide Area MLAT-WAM),
- Hassas pist gözetimi,
- Yükseklik gözetim birimi,

- Çevresel yönetim,
- Havaalanı operasyonları ve gelir yönetimidir [8,10-13].

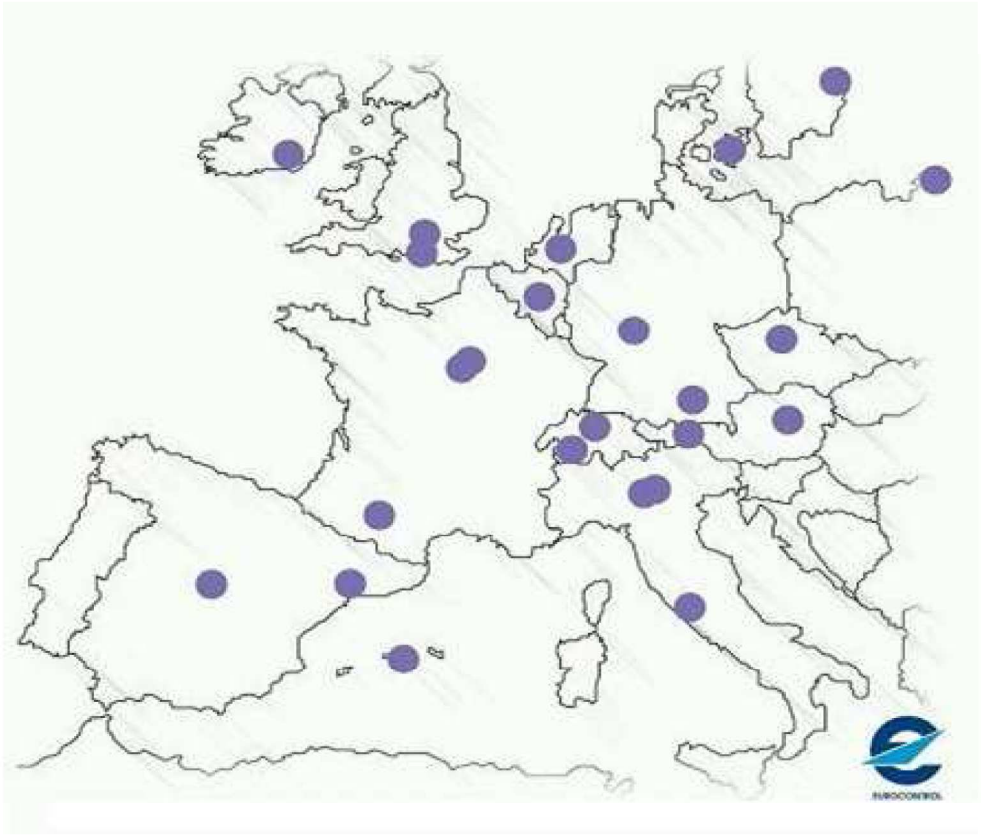
MLAT uygulamaları, ülkelerdeki seyrüsefer sağlayıcı kurumların hava sahalarını kapsama alanı olarak güncellerken, genişletirken ya da yeni sahalar oluştururken tamamen yeni bir şekilde düşünmelerini sağlamıştır. Çünkü SSR radarlarla kıyaslandığında kapsamaların gerçekten çok zor, çetin ve maliyetli olabileceği hava sahalarında bu tür sistemler, hem daha ucuz hem de daha kesin bilgiler verirler. Hatta radar kapsamalarında oluşabilecek boşluklar da bu sistemler sayesinde giderilebilir. Böylece uçaklar da daha rahat manevra yapabilirler [10-12].

Yani MLAT sistemler, sadece hava sahası kullanımını ve operasyonel verimliliği artıran sistemler olarak değil, aynı zamanda gözle görülür şekilde ekonomik fayda ve esneklik sağlayan sistemler olarak görülmelidirler. Resim 3.1'de örnek MLAT istasyonları gösterilmiştir.

Resim 3.1. Örnek MLAT istasyonları

3.1. Havaalanı Yüzeyi (LAM)

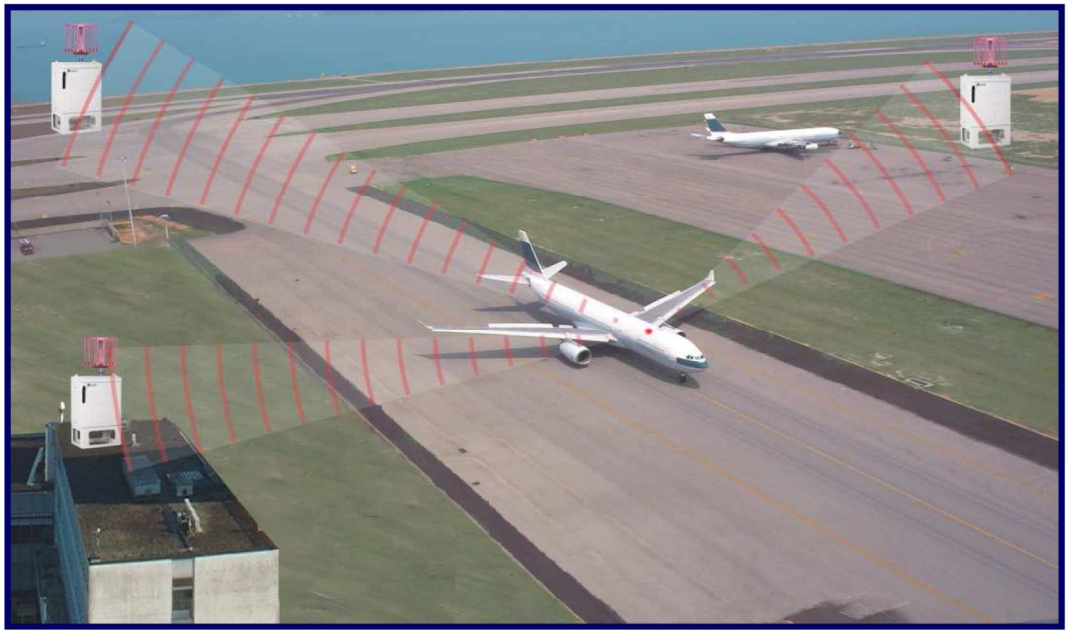
LAM (Lokal Alan MLAT; Local Areal MLAT) sistemleri de denilen bu sistemler genelde büyük havaalanları için geliştirilen A-SMGCS (Geliştirilmiş Yer Hareketleri Rehber ve Kontrol Sistemi; Advance Surface Movement and Ground Control System) çalışmalarının hızla ilerlemesi ile bu sistemlerin bir parçası olarak net bir resim elde etmek için SMR (Yüzey Hareket Radarı; Surface Movement Radar)'lerle birlikte kullanılırlar. Dünyada birçok büyük hava limanında halihazırda bu sistemler kullanılmaktadır ve ülkemizde de Esenboğa, Atatürk ve Antalya Havalimanlarına bu sistemler kurulma aşamasındadırlar [2]. Şekil 3.1'de 2007 yılına göre Avrupa'da MLAT sistemleri kullanılan havaalanları görülmektedir [11].



Şekil 3.1. 2007 yılına göre Avrupa'da MLAT kullanılan havaalanları

Söz konusu LAM sistemlerde uçakların havaalanı yüzeyindeki (pist, apron, taksi yolu) hareketlerinin izlenmesi, çarpışma ya da hadiselerin önlenmesi ile birlikte hızlı

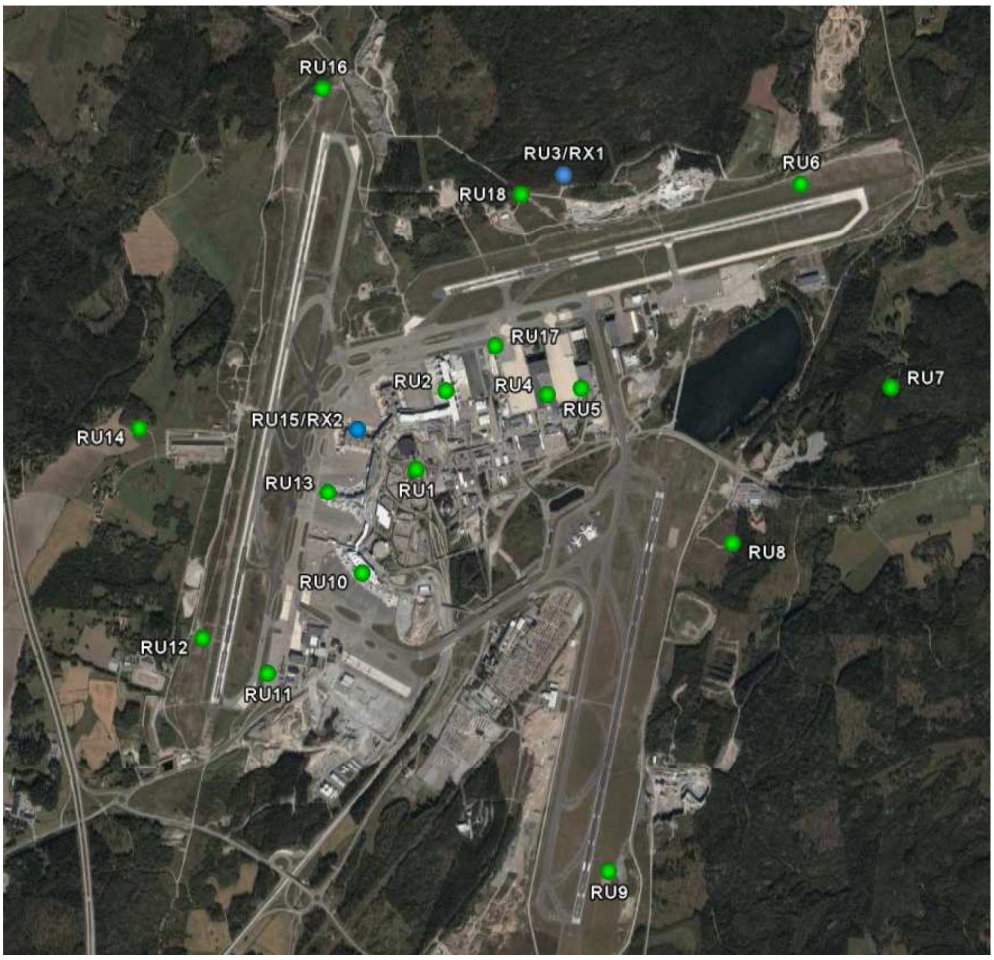
ve akışkan bir hava trafiği elde etmek amaçlanmıştır. Pozisyon doğruluğunun çok yüksek olması, havaalanlarında da bina ve benzeri engellerin çok olmasının sinyalleri etkilemesinden dolayı, genelde çok fazla alıcı anten kullanılır [10]. Havaalanı yüzeyi uygulamalarında uçaklar zaten yerde oldukları için uçakların 2 boyutlu pozisyon bilgisi elde edilir. Resim 3.2'de bir havaalanı yüzeyinde kullanılan MLAT sistem örnek olarak gösterilmiştir.



Resim 3.2. Havaalanı yüzeyi MLAT uygulaması

Yukarıda bahsedildiği gibi havaalanı yüzeyinde kullanılan MLAT sistemler, günümüzde A-SMGCS sistemlerinin büyük önem taşıyan elemanı olmuşlardır. Çünkü iyi çalışılmış olarak stratejik yerlere yerleştirilen MLAT istasyonları sayesinde, kontrol kulelerindeki hava trafik yer kontrolörleri, havaalanı yüzeyinin radar tarafından görülemeyen gizli bölgelerinin de bilgilerini meteoroloji koşullarına bağımlı olmaksızın kesin/doğru bir şekilde elde edebilmektedirler. Bu da hem hava trafik güvenliğini hem de hava trafik akış hızını artırır. Böylece havaalanlarının kapasitesi artırılabilmekte, uçaklar açısından maddi olarak da yakıt tasarrufu sağlanabilmektedir. Resim 3.3'te ise Stokholm Arlanda Havalimanı MLAT uygulaması örnek olarak görülmektedir. Burada yeşil olarak görülen noktalar alıcı

antenler ve yer istasyonları, mavi olarak görülen noktalar ise hem alıcı hem verici olarak kullanılan istasyonları göstermektedir [10]. Görüldüğü gibi üç adet pisti, taksi yollarını ve apronu kaplayabilmek için çok fazla istasyon kullanılmıştır.



Resim 3.3. Stokholm Arlanda Havalimanı MLAT uygulaması

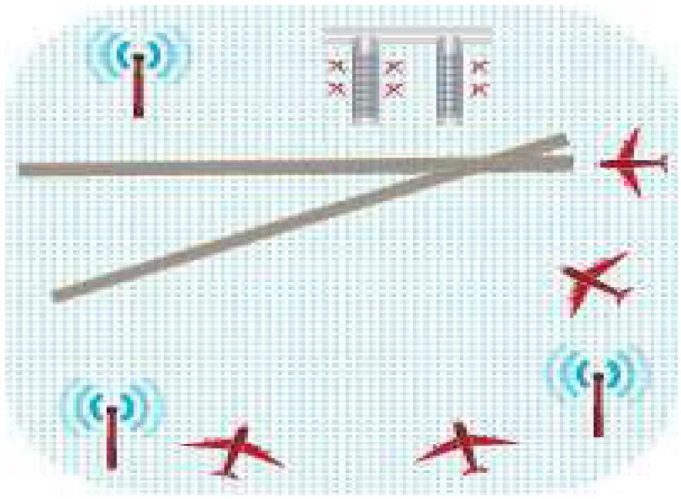
3.2. Terminal Alan

Dünyada birçok havaalanında, terminal alanlardaki (TMA; Terminal Area) düşük irtifa operasyonları, özellikle SSR radar sinyallerini etkileyen çok yüksek engellerle (bina, dağ vb.) sınırlandırılır. Bu da hava trafik kontrolörlerinin belirli bir irtifanın altını izleyememesi sorununu yaratır. Dolayısıyla uçakların artık iniş için yaklaşmalarını yaptıkları bu bölgelerdeki sınırlamalar hava trafiğini yavaşlatır.

Özellikle kötü hava koşullarında çok fazla geri dönme (divert) olayları yaşanması sebep olur.

Örneğin bu tür problemler Avusturya Inssbruck ve Çek Cumhuriyeti Ostrava havaalanlarında çokça yaşanmaktaydı. Innsbruck'te çevredeki dağlar minimum karar yüksekliğinin (MDA; Minimum Decision Altitute) havaalanının üstünde 31,000 ft olmasını zorlamaktaydı. Ostrava'da ise uçaklar gene dağlardan dolayı 6,000 ft'in altına inmekten özellikle kaçınmaktaydılar [10].

Bu havaalanları için bir çözüm bir veya daha çok SSR radarın havaalanı ya da yakınına kurulması olabilirdi. Fakat ekonomik analizler her iki seyrüsefer sağlayıcısı için ilgili bölgelere MLAT gözetim sistemi kurmanın daha az maliyetli ve operasyonel olarak daha avantajlı olduğunu göstermiştir [10]. MLAT'ın sadece kurulum, bakım ve kazanım açısından daha ucuz olmasından değil, aynı zamanda optimum terminal alan kapsaması sağlaması, daha hızlı ve daha doğru izleme sağlaması da onu avantajlı kılmıştır. Resim 3.4'te terminal alan MLAT uygulaması gösterilmiştir. Burada havaalanına yaklaşmaların pistlerin her tarafından olabileceği görülmektedir. Yani MLAT kapsaması sadece pist merkez hatlarını değil o havaalanı için daha önceden belirlenmiş ve yayınlanmış yaklaşma sektörünü de kapsayasacak şekilde olmalıdır.



Resim 3.4. Terminal alan MLAT uygulaması

3.3. Geniş Alan (WAM)

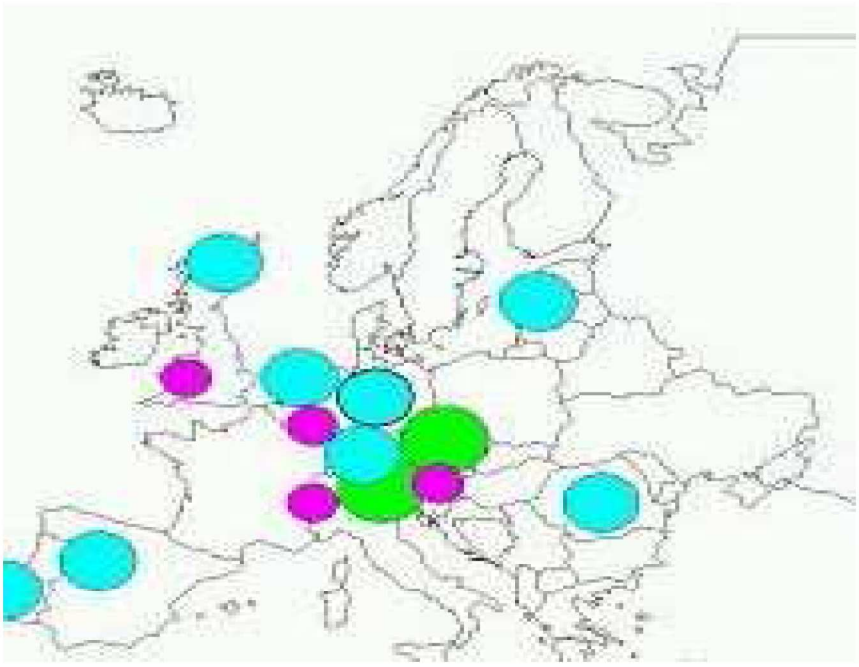
Hava trafiğinin dolayısıyla da gözetim ihtiyacının artmasıyla, klasik SSR radar kapdaması henüz yapılmamış bölgelerde, ya da yedekleme amaçlı ikincil/üçüncül kapsama gerektiren yerlerde seyrüsefer sağlayıcıları, yeni radar kurulumları yerine MLAT sistemlerin fayda/maliyet avantajlarından yararlanmak isterler.

Adından da anlaşılacağı üzere WAM (Wide Area MLAT)'da geniş hava sahalarının (50-200 NM) gözetimi amaçlanır ve bu sistemlerde istasyonlar birbirlerine yaklaşık 100 km kadar aralıklarla yerleştirilirler. Burada asıl hedef yol kontrol (en-route) amaçlı kullanımdır. Yani belirli irtifanın üstünde, havayollarında uçan uçakların gözetimi amaçlanmıştır. Amerika/Detroit, Kanada/Vancouver, Yeni Zelanda, Avustralya/Tazmanya, Çek Cumhuriyeti/Prag'da bu amaçlı kullanılan sistemler halihazırda mevcuttur [10]. Ayrıca ekonomik sebeplerden dolayı Amerika'da Kolorado, İngiltere'de Kuzey Denizi ve East Midlands bölgesine ve Tayvan'a kurulumlar planlanmıştır [10,13]. Bu tarz bölgelerde WAM, SSR'ye göre daha yüksek bir kapsama alanı, daha doğru iz takibi, önemli derecede maliyet avantajı ve çok daha erken operasyona geçebilme imkanı sağlar. Resim 3.5'te örnek bir WAM sistemi gösterilmiştir.

Resim 3.5. WAM uygulaması

Diğer taraftan SSR radar değişimine ihtiyaç duyulan bölgelerde de yeni bir SSR radar yerine WAM sistemleri kurulabilmektedir. Örneğin Ermenistan yaptığı maliyet/performans analiz çalışmaları sonucunda artık değişim zamanı gelmiş olan SSR radarının yerine WAM sistemi kurmanın daha avantajlı olduğunu görmüş ve bu sistemleri kurmuştur [10].

MLAT sistemleri kullanım düşüncesi her geçen gün gittikçe artmaktadır. Şekil 3.2'de Avrupa'da kullanılan mevcut sistemler yeşil daireler ile, yol kontrol amaçlı kurulması planlanan yerler açık mavi ile, terminal alan amaçlı kurulması planlanan yerler de mor renk ile gösterilmiştir [4].

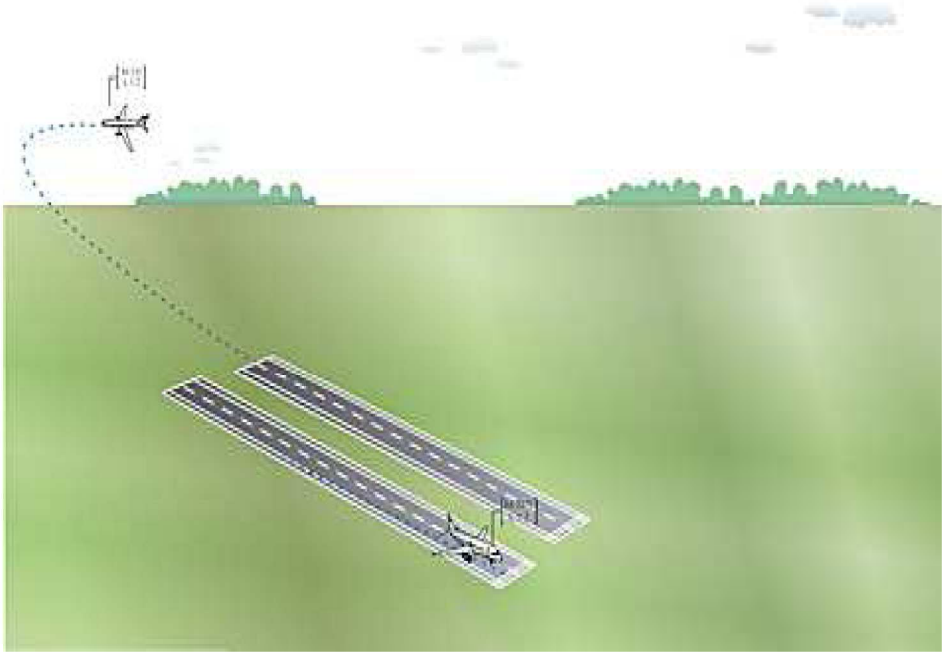


Şekil 3.2. MLAT kurulmuş/kullanılan ve kurulması planlanan yerler

3.4. Hassas Pist Gözetimi

MLAT uygulamalarının uçakların iniş kapasitesini önemli bir şekilde artırırken daha yüksek emniyet oranları sağlamakta olduğu kanıtlanmıştır.

Paralel pistleri olan havaalanlarında, uçaklar ayrı olarak atanmış pistlere doğru yakın rotalarda uçarlar. Fakat birçok havaalanında pistler birbirine çok yakın olduğu için uçakların yakın şekilde uçmasına güvenlik açısından izin verilmez. İki pist merkez hattı arası uzaklık 760 m'den küçükse pistler birbirine bağımlı pist olmuş olur, yani iki piste aynı anda uçak indirip kaldırılmaz (Resim 3.6) [10].

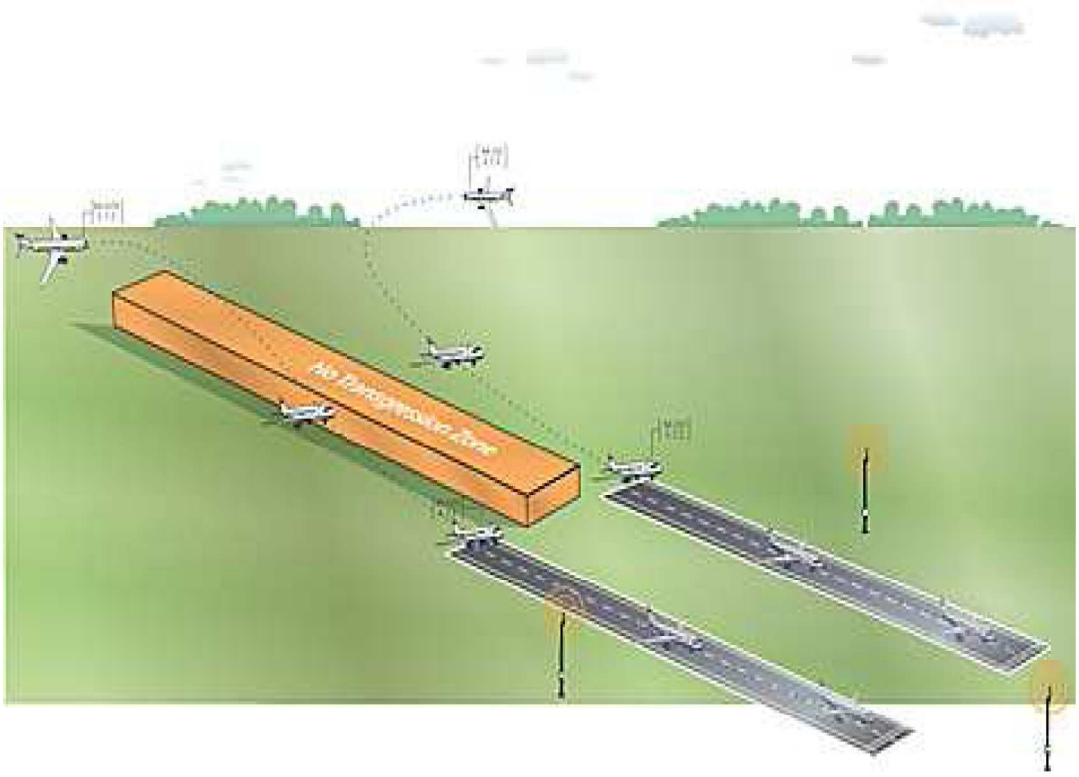


Resim 3.6. Hassas pist gözetimi olmadan pistlerin kullanımı

Dolayısıyla özellikle kötü hava koşullarında uçaklar arasındaki ayırma oranları artırılır, havaalanı kapasitesi azalır ve pistler aynı zamanda kullanılamaz. Bu duruma çözüm olarak hassas pist gözetimi amaçlı, dönmeden elektronik tarama yapan anten dizilerinden oluşan, çok yüksek azimutu olan monopuls SSR radar geliştirilmiştir (Resim 3.7) .

Resim 3.7. Hassas pist gözetim radarı

Fakat bu tür sistemlerin kurulum ve bakım maliyetleri çok yüksek olduğu için kabul edilebilirliği düşüktür. Bu tür elektronik tarama (e-scan) yapan radarlar maliyet açısından sıkıntı yaratırken, MLAT'ın hassas pist gözetimi için oldukça düşük maliyetlerle ve gerekli tüm ihtiyaçları fazlasıyla karşılayabilecek şekilde kullanılabileceği gösterilmiştir. MLAT hassas pist gözetim sistemlerinin özellikle kötü hava koşullarında ve yoğun hava trafiğinde kapasiteyi %30 ya da daha fazla artırdığı rapor edilmiştir. Bu tür faydalar düşünüldüğünde Pekin, Kuala Lumpur, Sidney ve Ditroit'teki kurulu sistemler dışında dünyada yakın zamanda MLAT hassas pist gözetimi sistemlerinin artarak kurulacağı tahmin edilmektedir [10,13]. Resim 3.8'de MLAT hassas pist gözetimi ile pistlerin kullanımı, aynı anda uçak indirip kaldırabilme hadisesi gösterilmiştir.



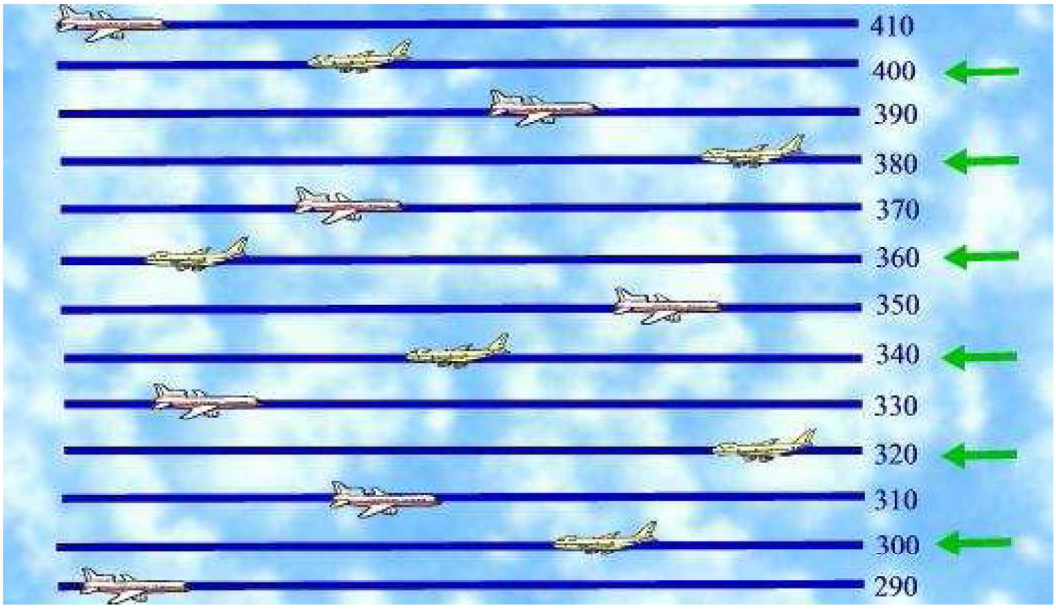
Resim 3.8. MLAT hassas pist gözetimi ile pistlerin kullanımı

3.5. Yükseklik Gözetim Birimi

1980'lerden sonra yüksek irtifalara çıkabilen jet uçakların çoğalmasıyla, yüksek irtifalardaki hava trafiği giderek artmıştır ve özellikle 29 000-41 000 ft arası, uçaklar için yakıt maliyetleri açısından tercih edilen irtifalar olmuştur [2]. Bununla birlikte 29 000 ft'in üstünde, düşen hava yoğunluğu nedeniyle geleneksel basınç altimetreleri daha yanlış sonuç vermektedirler [10]. Dolaysıyla 29 000 ft'in altında uçan uçaklar arasında dikey 1000 ft'lik, üstünde uçan uçaklar arasında ise güvenlik nedeniyle dikey 2000 ft'lik ayırma uygulanması gerekmekteydi [2].

1990'lı yıllarda geliştirilmeye başlayan yeni altimetre teknolojisi ile irtifa ölçümlerinde çok önemli gelişmeler sağlanmış ve dünya genelinde RVSM (Azaltılmış Dikey Ayırma Minimumu; Reduced Vertical Seperation Minima) başlamasıyla 41 000 ft'e kadar dikey ayırma minimumu normal değer olan 1000 ft

olmuştur. Böylece 29 000 ft-41 000 ft arasında 6 yeni hava koridoru açılabilmiştir (Resim 3.9) [2].



Resim 3.9. RVSM

Fakat RVSM hava seviyelerinde uçmak isteyen bütün uçaklar, yeni gerekliliklere uyum sağlamak için yeni ekipmanlarla donatılmışlar ve periyodik olarak bu ekipmanların uygun ve belirlenmiş toleranslarda çalıştığını doğrulamak zorunda kalmışlardır.

MLAT söz konusu çalışmalarda en iyi teknik olarak seçilmiştir ve bu amaçla geliştirilmiş sistemler dünyanın birçok noktasına kurulmuş ve çalışmaktadır. Bu sistemler klasik SSR'larda elde edilemeyen doğru yükseklik ölçüm bilgisini sağlayabilmektedirler. Uçaklarda Mod C ile gelen bilginin doğrulaması, bu MLAT sistemler ile yapılmaktadır [10].

3.6. Çevresel Yönetim

Günümüzde birçok havaalanları gürültü azaltma prosedürlerine kuvvetli bir şekilde bağlıdırlar. Fakat havaalanı yakınındaki yerleşim birimlerinden gürültü ile ilgili

gelen şikayetler havaalanı işleticileri için sürekli devam eden sorunlardır ve genelde bu gürültüye sebep olan uçağı bulmak gerçekten çok zor olmaktadır.

MLAT sistemlerin ilk uygulamalarından biri de bu fonksiyonu yerine getirmekti. Sistemler tüm inen ve kalkan uçakları kendi kesin rotalarında kaydetmekte ve bu rota üzerindeki herhangi bir noktadan hangi zamanda geçtiğini göstermekteydi. Bu kayıtlar da yargı için yasal kanıt olarak kabul edilmişti [10].

Diğer çevresel düzenlemeler etkili olmaya başladıkça MLAT, havaalanı yönetimi için ses, emisyon ve diğer uçuş operasyon verileri konusunda, çok etkili ve hızla ulaşılabilir veriler sağlayabilmektedir.

3.7. Havaalanı Operasyonları ve Gelir Yönetimi

Havaalanlarında uçakların ve araçların hareketleri sabit olarak ölçülmeli ve analiz edilmelidir. Çünkü havaalanı yüzeyindeki herhangi bir aksaklık o havaalanı operasyonlarını domino etkisi gibi felce uğratabilmektedir.

MLAT sistemler havayolu ve havaalanı personeline gerçek zamanlı havaalanı durum bilgisi ve uçak izi bilgisi sağlayarak, kapı yönetiminin geliştirilmesi ve apron hareketlerinin düzenlenmesi için havaalanı kaynaklarının planlanması ve zamanlanması olan CDM (İşbirlikçi Karar Yapımı; Colloborative Decision Making) ‘yi desteklerler [10].

Normalde, havaalanı işleticileri uçak şirketlerinin ya da uçakların, iniş, park, kapı kullanımı gibi havaalanlarının en önemli gelir kaynakları için hazırladıkları kendi raporlarına güvenmektedirler. Bu da, havaalanları için doğrulama yapacak sistemler yoksa gelir kayıplarına sebep olmaktadır.

MLAT sistemler otomatik olarak zamansal ve doğru faturalar çıkarmak için uçuş izi ve kimlik verisini sağlayabilmektedirler.

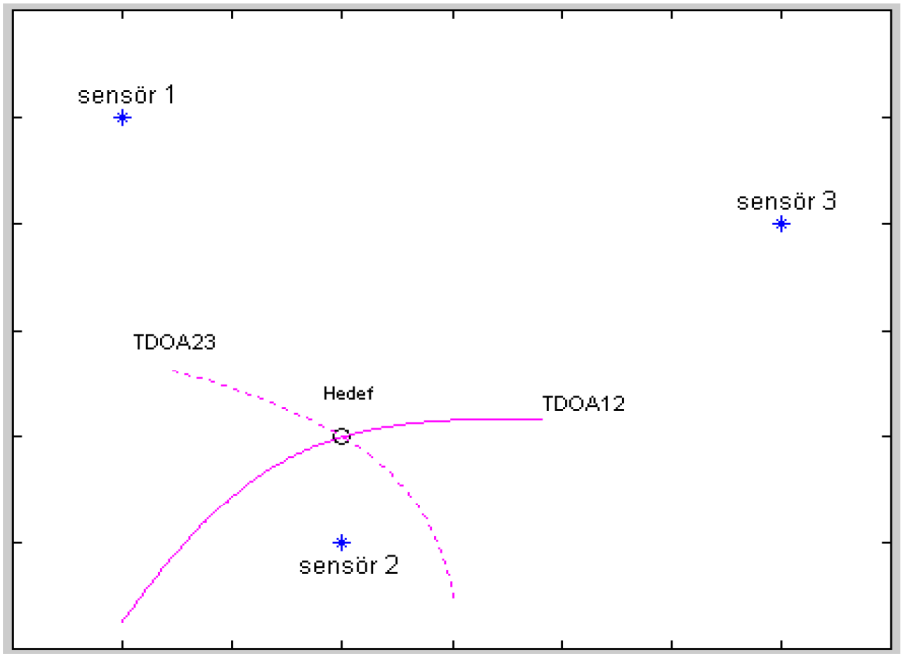
4. TDOA TEKNİĞİ VE EN KÜÇÜK KARELER YÖNTEMİ

4.1. TDOA (Ulaşım Zamanı Farkı)

TDOA tekniği bu çalışmada iki boyutlu ve üç boyutlu olarak ele alınmış ve simüle edilmiştir.

4.1.1. İki boyutlu sistem

Hedef verici en az üç sensör tarafından alınabilecek sinyalleri gönderir. Her bir sensör gelen sinyalin ulaşım zamanını (TOA; Time of Arrival) kaydeder. İki sensör arasındaki ulaşım zamanı farkının (TDOA; Time Difference of Arrival) oluşturduğu hiperbol üzerindeki herhangi bir noktada sinyali gönderen söz konusu verici bulunur [8]. Şekil 4.1'de bu durum gösterilmiştir.



Şekil 4.1. Hedef, alıcı sensörler ve hiperboller

TDOA tekniğinin temeli iki nokta arasındaki uzaklık formülüdür. Eş. 4.1'de iki nokta arasındaki uzaklık formülü gösterilmiştir.

$$d = \sqrt{(x - x_1)^2 + (y - y_1)^2} \tag{4.1}$$

Bilinmeyen bir yerde (x,y) bir vericimiz olduğunu düşünelim. Ayrıca bilinen 3 noktada da MLAT alıcılarımız olduğunu düşünelim: Merkez, sol ve sağ konumlar.

Vericiden (x,y) yayılan sinyallerin her bir alıcı konumuna ulaşma zamanı mesafenin sinyal yayılım hızına (c) bölümüdür: Eş. 4.2, Eş. 4.3 ve Eş. 4.4'te sırasıyla sol, sağ ve merkez alıcılar için sinyalin ulaşma zamanı gösterilmiştir.

$$T_L = \frac{1}{c}(\sqrt{(x - x_L)^2 + (y - y_L)^2} \tag{4.2}$$

$$T_R = \frac{1}{c}(\sqrt{(x - x_R)^2 + (y - y_R)^2} \tag{4.3}$$

$$T_M = \frac{1}{c}(\sqrt{(x - x_M)^2 + (y - y_M)^2} \tag{4.4}$$

Eğer M konumu koordinat sisteminin orijini kabul edilirse, T_M Eş. 4.5'teki gibi yazılır.

$$T_M = \frac{1}{c}(\sqrt{x^2 + y^2}) \tag{4.5}$$

Daha sonra merkez konuma direkt ulaşan ve kenar konumlardan gelen sinyallerin ulaşım zamanı farkı Eş. 4.6 ve Eş. 4.7'deki gibi olur.

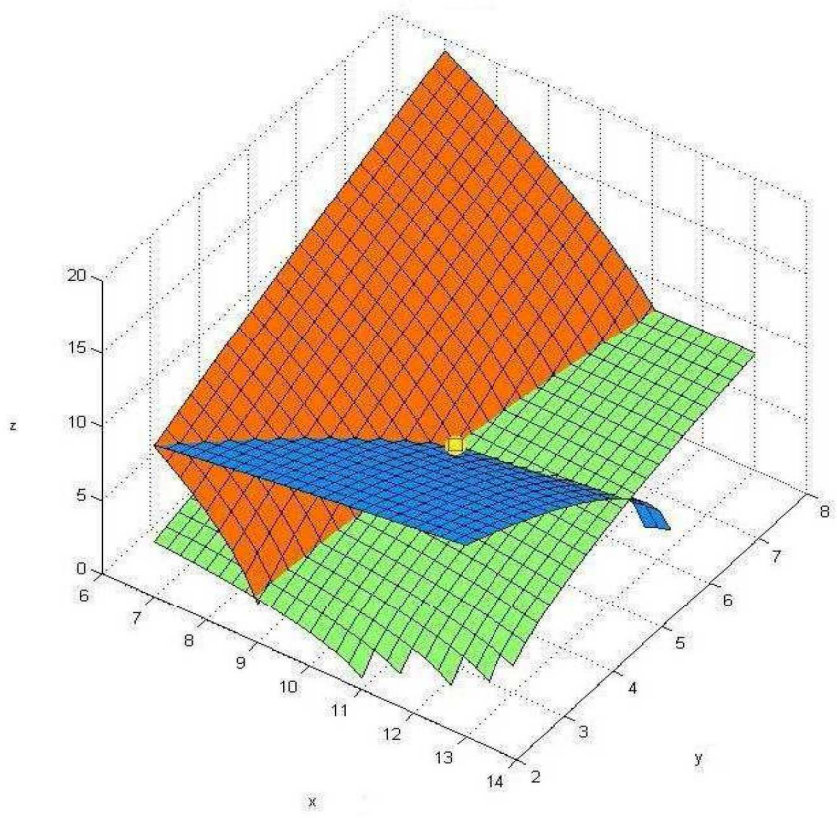
$$\tau_L = T_L - T_M = \frac{1}{c}(\sqrt{(x - x_L)^2 + (y - y_L)^2} - \sqrt{x^2 + y^2}) \tag{4.6}$$

$$\tau_R = T_R - T_M = \frac{1}{c}(\sqrt{(x - x_R)^2 + (y - y_R)^2} - \sqrt{x^2 + y^2}) \tag{4.7}$$

Burada (x_L, y_L) sol alıcı konumunu, (x_R, y_R)'de sağ alıcı konumunu göstermektedir. Ayrıca c'de sinyalin yayılım hızı, genelde de ışık hızıdır [5, 8]. Her bir eşitlik farklı bir hiperbolu gösterir.

4.1.2. Üç boyutlu sistem

Üç boyutlu sistem dediğimizde iki boyuta ek olarak üçüncü boyut olan z boyutunu da düşünmemiz gerekir. Bu durumda vericiden yayılan sinyallerin oluşturduğu zaman farkı artık hiperbol değil hiperboloid şeklinde olacaktır (Şekil 4.2) [1].



Şekil 4.2. Hiperboloidler ve kesişimi

Bilinmeyen ve bulmak istediğimiz bir konumda (x,y,z) verici olduğunu düşünelim. Ayrıca dört alıcıdan oluşan ve koordinatlarını bildiğimiz bir MLAT sistemi düşünelim. (merkez, sol, sağ ve dördüncü konum)

Aynı iki boyutlu sistemde olduğu gibi vericiden (x,y,z) yayılan sinyallerin her bir alıcı konumuna ulaşma zamanı mesafenin sinyal yayılım hızına (c) bölümüdür ve sırasıyla Eş. 4.8, Eş. 4.9, Eş. 4.10 ve Eş. 4.11'de gösterildiği gibidir.

$$T_L = \frac{1}{c}(\sqrt{(x-x_L)^2 + (y-y_L)^2 + (z-z_L)^2} \quad (4.8)$$

$$T_R = \frac{1}{c}(\sqrt{(x-x_R)^2 + (y-y_R)^2 + (z-z_R)^2} \quad (4.9)$$

$$T_Q = \frac{1}{c}(\sqrt{(x-x_Q)^2 + (y-y_Q)^2 + (z-z_Q)^2} \quad (4.10)$$

$$T_M = \frac{1}{c}(\sqrt{(x-x_M)^2 + (y-y_M)^2 + (z-z_M)^2} \quad (4.11)$$

Eğer M konumu koordinat sisteminin orijini kabul edilirse Eş. 4.12 elde edilir.

$$T_M = \frac{1}{c}(\sqrt{x^2 + y^2 + z^2}) \quad (4.12)$$

Daha sonra merkez konuma direkt ulaşan ve kenar konumlardan gelen sinyallerin ulaşım zamanı farkı Eş. 4.13, Eş. 4.14 ve Eş. 4.15'te gösterildiği gibi olur.

$$\tau_L = T_L - T_M = \frac{1}{c}(\sqrt{(x-x_L)^2 + (y-y_L)^2 + (z-z_L)^2} - \sqrt{x^2 + y^2 + z^2}) \quad (4.13)$$

$$\tau_R = T_R - T_M = \frac{1}{c}(\sqrt{(x-x_R)^2 + (y-y_R)^2 + (z-z_R)^2} - \sqrt{x^2 + y^2 + z^2}) \quad (4.14)$$

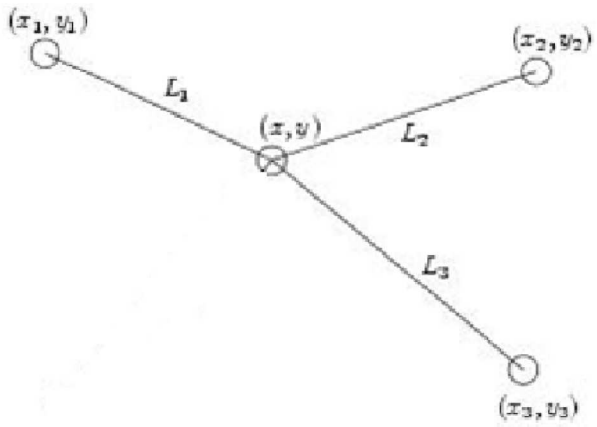
$$\tau_Q = T_Q - T_M = \frac{1}{c}(\sqrt{(x-x_Q)^2 + (y-y_Q)^2 + (z-z_Q)^2} - \sqrt{x^2 + y^2 + z^2}) \quad (4.15)$$

Burada (x_L, y_L, z_L) sol alıcı konumunu, (x_R, y_R, z_R) sağ alıcı konumunu, (x_Q, y_Q, z_Q)'de diğer bir alıcı konumunu göstermektedir. Ayrıca c'de sinyalin yayılım hızı, genelde de ışık hızıdır. Her bir eşitlik farklı bir hiperboloidi gösterir [8].

4.2. En Küçük Kareler Yöntemi

En küçük kareler yöntemi, birbirine bağlı olarak değişen iki fiziksel büyüklük arasındaki matematiksel bağlantıyı, mümkün olduğunca gerçeğe uygun bir denklem olarak yazmak için kullanılan, standart bir regresyon yöntemidir. Bir başka deyişle bu yöntem, ölçüm sonucu elde edilmiş veri noktalarına "mümkün olduğu kadar yakın" geçecek bir fonksiyon eğrisi bulmaya yarar.

Şekil 4.3'teki gibi bir problemi ele alalım. Burada ölçtüğümüz üç adet uzaklık bize üç adet vektör verir. Bunlar $\bar{L} = (L_1, L_2, L_3)$'tür.



Şekil 4.3. En küçük kareler yöntemi problemi

$\overline{X}$ pozisyon değişkenlerini içeren vektör, $\overline{L}$ yapılan ölçümleri içeren vektör ise Eş. 4.16 yazılabilir [9].

$$A\overline{X} = \overline{L} \tag{4.16}$$

Eş. 4.16'da A, ölçümlerimizi elde etmek için $\overline{X}$ vektöründe kullanılan tasarım matrisidir. Ölçüm cihazlarının hassaslığındaki sınırlamalardan dolayı gereğinden fazla yapılan fiziksel ölçümler tutarsız sonuçlara yol açar. Bu da tek bir sonuç olmadığını gösterir, burada bizim yapabileceğimiz tek bir sonuca yaklaşmaya çalışmaktır ve kullanacağımız metot ise en küçük kareler yöntemidir.

Sonuç vektörüne ilk yaklaşım $\overline{X_0}$ olursa, $\bar{X}$ vektörü Eş. 4.17'deki gibi yazılabilir.

$$\bar{X} = \overline{X_0} + \bar{X} \tag{4.17}$$

Bütün ölçümler temel olarak kesin olmadığı için, ölçüm değerleri olan L uzunlukları kararsız sonuçlar verecektir. Dolayısıyla kararsızlıkları gidermek için ölçümlere artan vektörü $\bar{V}$ eklenir [9]. Böylece Eş. 4.16'daki $\bar{L}$ Eş.4.18'deki gibi yazılabilir.

$$\bar{L} = \bar{L} + \bar{V} \tag{4.18}$$

En küçük kareler yöntemi, en iyi $\overline{X}$ yaklaşımının, kararsızlıkların toplamının karesini minimize eden yaklaşım olduğunu gösterir, $V^T V$'yi minimize eden yaklaşım gibi.

4.2.1 Ağırlık matrisi

Ölçümlerimizin her biri (L elemanlarından oluşan) aynı doğruluğa sahip olmayacaktır. Bu her bir ölçüme bilinen bir ağırlık matrisi verilerek açıklanabilir. P elemanları bu ağırlıklar olan matristir [9]. Bu durumda en küçük kareler ölçütü Eş. 4.19'daki gibi olur.

$$\overline{V^T} P \bar{V} = minimum \tag{4.19}$$

4.2.2. Doğrusallaştırma

$F(\overline{X}) = \overline{L}$ olduğunu ve F'nin A matrisi tarafından temsil edilen $\overline{X}$'in bir fonksiyonu olduğunu biliyoruz (Eş. 4.16). Bu denklemi Eş. 4.20'deki gibi tekrar yazabiliriz:

$$F(\overline{X_0} + \bar{X}) = \bar{L} + \bar{V} \tag{4.20}$$

Toplam ölçülen vektörler ve sonuç vektörleri ($\overline{L}$ ve $\overline{X}$) arasındaki ilişkiyi tanımlayan matematiksel modeller Eş. 4.21'deki gibi genel forma sahiptir.

$$F(\bar{X}, \bar{L}) = 0 \tag{4.21}$$

Bu modelleri doğrusallaştırma, doğrusal olmayan F fonksiyonlarını kendi Taylor serisi doğrusal yaklaşımlarıyla değiştirerek, yani çözüm vektörüne ilk yaklaşım noktasını (X_0) ve ölçüm vektörünün ($\bar{L}$) ölçülmüş değerlerini genişleterek yapılabilir [9].

Eş. 4.17 ve Eş. 4.18'i kullanarak yapacağımız doğrusallaştırmadan Eş. 4.22 elde edilir.

$$F(\bar{X},\bar{L}) = F(\overline{X_0},\bar{L}) + \frac{\partial F(\overline{X}=\overline{X_0},\bar{L}=L)}{\partial \overline{X}}\overline{X} + \frac{\partial F(\overline{X}=\overline{X_0},\bar{L}=\bar{L})}{\partial \bar{L}}\overline{V} = 0 \qquad (4.22)$$

Bu da Eş. 4.23'teki gibi yazılabilir.

$$\bar{W} + A\bar{X} + B\bar{V} = 0 \qquad (4.23)$$

Eş. 4.23'te $\overline{W} = F(\overline{X_0},\bar{L})$, $A = \frac{\partial F(\overline{X}=\overline{X_0},\bar{L}=\bar{L})}{\partial \overline{X}}$ ve $B = \frac{\partial F(\overline{X}=X_0,\bar{L}=\bar{L})}{\partial \bar{L}}$ olarak verilmektedir.

Şekil 4.3'teki problemde ölçülmüş değerler $\overline{X}$ parametrelerinin fonksiyonları olarak açıkça gösterilebilirler ($F(\overline{X}) = \overline{L}$ olduğunu hatırlayalım).

Bu durumda doğrusallaştırmadan Eş. 4.24 elde edilir [9].

$$F(\overline{X}) - \bar{L} = F(\overline{X_0}) + \frac{\partial F(\overline{X}=X_0)}{\partial \overline{X}}\overline{X} - (\bar{L}+\overline{V}) = 0 \qquad (4.24)$$

$\bar{L}$

Ya da

$$\overline{W} + A\overline{X} - \overline{V} = 0 \qquad (4.25)$$

Burada $\overline{W} = F(\overline{X_0}) - \bar{L}$ ve $A = \frac{\partial F(\overline{X}=\overline{X_0})}{\partial \overline{X}} = \begin{pmatrix} \frac{\partial F_1}{\partial X_1} & \frac{\partial F_1}{\partial X_2} & \bullet & \bullet \\ \frac{\partial F_2}{\partial X_1} & \bullet & & \\ \bullet & & \bullet & \\ \bullet & & & \bullet \end{pmatrix}$ 'dir.

Şimdi problemimiz

$\overline{V^T} P \overline{V}$ = minimum

X (bastırılmaya bağlı olan): $\overline{W} + A\overline{X} - \overline{V} = 0$ olmuştur. Bu bastırılmayı gerçekleştirmek için Lagranj çarpanları kullanırız.

4.2.3. Lagranj çarpanları

$f_2(x, y) = 0$ bastırılmasına bağlı olan $f_1(x, y)$ fonksiyonunu minimize etmek istersek lagranj metodu kullanırız ve bu da 3 aşamadan oluşur [9].

1) Değişim fonksiyonu oluşturulur (Eş. 4.26).

$$\phi = f_1(x, y) + k f_2(x, y) \tag{4.26}$$

Burada k Lagrange çarpanı olarak bilinen belirsiz sabittir.

2) Değişim fonksiyonunun türevleri sıfıra eşitlenir (Eş. 4.27).

$$\frac{\partial \phi}{\partial x} = 0, \ \frac{\partial \phi}{\partial y} = 0 \tag{4.27}$$

3) Bu üç denklem çözülür.

$$f_2(x, y) = 0, \ \frac{\partial \phi}{\partial x} = 0, \ \frac{\partial \phi}{\partial y} = 0$$

Böylece değişim fonksiyonu aşağıdaki gibi oluşturulabilir.

$$\phi = \overline{V^T} P \overline{V} + 2\overline{K^T}(A\bar{X} - \bar{V} + \bar{W}) \tag{4.28}$$

Burada $\overline{K}$ Lagranj çarpanlarının kolon vektörüdür ve uyum için 2 ile çarpılmıştır [9]. Türevler sonucunda Eş. 4.29 elde edilir.

$$\frac{\partial \phi}{\partial \overline{V}} = 2\overline{V^T}P - 2\overline{K^T} = 0 \tag{4.29}$$

Eş. 4.29 indirgenerek Eş. 4.30 elde edilir [9].

$$P\overline{V} - \overline{K} = 0 \tag{4.30}$$

ve

$$\frac{\partial \phi}{\partial \overline{X}} = 2\overline{K^T}A = 0 \tag{4.31}$$

yani

$$A^T\overline{K} = 0 \tag{4.32}$$

Sonuç olarak çözmek istediğimiz üç eşitlik Eş. 4.25, Eş. 4.30 ve Eş. 4.32 olur.

$$\overline{W} + A\overline{X} - \overline{V} = 0$$
$$P\overline{V} - \overline{K} = 0$$
$$A^T\overline{K} = 0$$

Bu eşitlikler tek bir hipermatris şeklinde yazılabilirler (hipermatrislerin kendi elemanları da birer matristirler) [9].

$$\begin{pmatrix} P & -I & 0 \\ -I & 0 & A \\ 0 & A^T & 0 \end{pmatrix} \begin{bmatrix} \overline{V} \\ \overline{K} \\ \overline{X} \end{bmatrix} + \begin{bmatrix} 0 \\ \overline{W} \\ 0 \end{bmatrix} = 0 \tag{4.33}$$

Bu eşitlikler sonuç olarak aşağıdaki eşitliklere indirgenirler [9].

$$\overline{X} = -(A^T PA)^{-1} A^T P\overline{W} \tag{4.34}$$

$$\overline{K} = P(A\overline{X} + \overline{W}) \tag{4.35}$$

$$\overline{V} = P^{-1}\overline{K} = A\overline{X} + \overline{W} \tag{4.36}$$

Burada A, P matisleri ve $\overline{W}$ vektörü bilinirse $\overline{X}$ de çözülebilir.

5. FONKSİYONLAR, PROGRAMLAR VE AÇIKLAMALARI

Bu tezde bahsedilen simülasyon çalışmalarını gerçekleştirmek için MATLAB programlama dili kullanılmıştır. Aşağıda yazılmış olan fonksiyonlar ve programların açıklamaları bulunmaktadır.

5.1. İki Nokta Arasındaki Uzaklık

Burada kullandığımız F(a,b) fonksiyonu koordinatlardan oluşan iki vektörü alıp bunların arasındaki farkı bulur. Bu da ab=(b-a) vektörünü bulup onu transposesi ile çarparak elde edilebilir. Kod şöyledir:

```
function out = F(X1,X2)

out=sqrt((X1-X2)'*(X1-X2));
```

5.2. Çizim Fonksiyonları

Çizim fonksiyonları iki boyutlu ve üç boyutlu çalışma için ayrı ayrı yazılmıştır. Söz konusu fonksiyonlar kendi içlerinde de bazı fonksiyonlar içerirler.

5.2.1. İki boyutlu (hyperbolplot)

İki boyutlu örnek için oluşturulan heyperbolplot(x1,x2,x3,y1,y2,y3,X) çizim fonksiyonu ana kodumuz olan simulation2 tarafından çağrılır. Burada amaç anılan simülasyona girilen ve hesaplanan değerler sonucunda her bir alıcıya ulaşan sinyallerin birer hiperbol oluşturduğunu göstermek ve çizdirmektir. İki boyutlu çalışmada söz konusu hiperbollerin kesişim noktası bize vericimizin yerini göstermektedir (uçak gerçek pozisyonu). Bununla birlikte hyperplot fonksiyonunda Bölüm 4.1'de anlatılan TDOA tekniğiyle oluşturulan denklemler kullanılmıştır. Oluşturduğumuz kod aşağıdaki gibidir:

```
function hyperbolplot (x1,x2,x3,y1,y2,y3,X)

c=582749918.861987;%ışık hızı Nm/h

tdoa1=(sqrt((X(1)-x1).^2+(X(2)-y1).^2)-sqrt((X(1).^2+X(2).^2)))./c;
tdoa2=(sqrt((X(1)-x2).^2+(X(2)-y2).^2)-sqrt((X(1).^2+X(2).^2)))./c;
tdoa3=(sqrt((X(1)-x3).^2+(X(2)-y3).^2)-sqrt((X(1).^2+X(2).^2)))./c;

tdoa=[tdoa1,tdoa2,tdoa3];

syms x y ; h =(sqrt((x-x1)^2+(y-y1)^2)-sqrt(x^2+y^2))./c-tdoa(1); clf; % Clear old
figure
ezplot (h); axis equal;

syms x y ; h=(sqrt((x-x2)^2+(y-y2)^2)-sqrt(x^2+y^2))./c-tdoa(2); hold on; % Clear
old figure
ezplot(h); axis equal;

syms x y ; h=(sqrt((x-x3)^2+(y-y3)^2)-sqrt(x^2+y^2))./c-tdoa(3); hold on; % Clear
old figure
ezplot(h); axis equal;grid on
hold on; plot(X(1),X(2),'r*')
```

Kodda görüleceği üzere iki boyutlu çizimleri yaptırırken ezplot fonksiyonu kullanılmıştır. Anılan fonksiyon kendi denklemlerimizi oluşturup çizim yaptırabilmemiz için bize kolaylık sağlamaktadır ve kodu Ek-1'de verilmiştir.

5.2.2. Üç boyutlu (hyperboloidplot)

Üç boyutlu çizimler için ise hyperboloidplot(x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,X) fonksiyonu kullanılmıştır. Üçüncü boyuta geçtiğimizde artık hiperboller hiperboloidlere dönüştürülür ve z (yükseklik) koordinatları da işin içine girer. Söz

konusu yükseklik değeri bize uçaktan mod C bilgisiyle direkt olarak gelmektedir. Dolayısıyla iki boyutta yapılan ufak değişikliklerle hyperboloidplot fonksiyonu oluşturulmuştur. Kod aşağıda verilmiştir:

```
function hyperboloidplot (x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,X)

c=582749918.861987;%ışık hızı Nm/h

tdoa1=(sqrt((X(1)-x1).^2+(X(2)-y1).^2+(X(3)-z1).^2)-
sqrt((X(1).^2+X(2).^2+X(3).^2))./c;

tdoa2=(sqrt((X(1)-x2).^2+(X(2)-y2).^2+(X(3)-z2).^2)-
sqrt((X(1).^2+X(2).^2+X(3).^2))./c;

tdoa3=(sqrt((X(1)-x3).^2+(X(2)-y3).^2+(X(3)-z3).^2)-
sqrt((X(1).^2+X(2).^2+X(3).^2))./c;

tdoa4=(sqrt((X(1)-x4).^2+(X(2)-y4).^2+(X(3)-z4).^2)-
sqrt((X(1).^2+X(2).^2+X(3).^2)))./c;
 tdoa=[tdoa1,tdoa2,tdoa3,tdoa4];

syms x y z; h=(sqrt((x-x1)^2+(y-y1)^2+(z-z1)^2)-sqrt(x^2+y^2+z^2))/c-tdoa(1); clf;
% Clear old figure
ezimplot3(h,'b'); axis equal

syms x y z; h=(sqrt((x-x2)^2+(y-y2)^2+(z-z2)^2)-sqrt(x^2+y^2+z^2))/c-tdoa(2);
hold on; % Clear old figure
ezimplot3(h,'g'); axis equal

syms x y z; h=(sqrt((x-x3)^2+(y-y3)^2+(z-z3)^2)-sqrt(x^2+y^2+z^2))/c-tdoa(3);
hold on; % Clear old figure
ezimplot3(h,'k');
```

```
syms x y z; h=(sqrt((x-x4)^2+(y-y4)^2+(z-z4)^2)-sqrt(x^2+y^2+z^2))./c-tdoa(4);
ezimplot3(h,'y');

hold on; plot3(X(1),X(2),X(3),'r*')
axis equal;grid on
```

Burada çizimleri yaptırırken kullanılan fonksiyon ise ezimplot3 fonksiyonudur. Ezplot fonksiyonundan farklı olarak üç boyutlu denklemler için grafik çizdirmeye yarayan fonksiyondur. Anılan ezimplot3 fonksiyonunun kodunda yaptığımız değişikliklerle her bir hiperboloidin farklı renkte çizdirilmesi ve çizimlerin daha net bir şekilde gösterilmesi sağlanmıştır.

5.3. Yer Kestirim Fonksiyonları

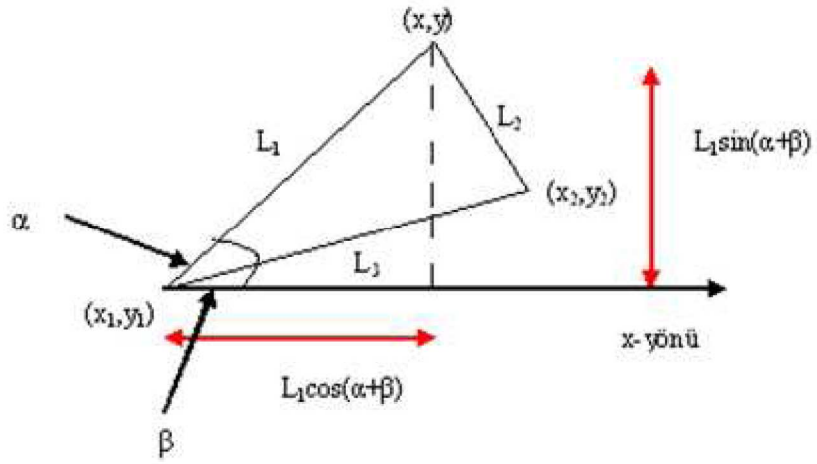
Yer kestirim fonksiyonları da çizim fonksiyonlarında olduğu gibi iki ve üç boyutlu olarak hazırlanmıştır. Dolayısıyla bu fonksiyonları da iki ve üç boyutlu olarak ayrı ayrı incelemek gerekir.

5.3.1 İki boyutlu yaklaşım (locate2)

İki boyutlu yer kestirim fonksiyonu locate2(x1,y1,L1,x2,y2,L2,x3,y3) verilen üç alıcı koordinatı ile bu koordinatlardan elde ettiğimiz iki ölçümü kullanarak bulmak istediğimiz noktanın koordinatlarını hesaplayan fonksiyondur.

Öncelikle locate2 fonksiyonu satır olarak verilen x, y koordinatlarını (örneğin (2,5)), sütunlara (2;5) çevirir. Böylece bu koordinatları iki nokta arasındaki uzaklığı bulmak için oluşturduğumuz F fonksiyonu ile kullanabilmekteyiz.

Daha sonra fonksiyon iki alıcı noktasına ve bulmak istediğimiz noktaya odaklanır. Örneğin Şekil 5.1'e bakarsak (x_1, y_1) ve (x_2, y_2) iki alıcı noktamız ve (x, y)'de bulmaya çalıştığımız verici noktamız olsun.



Şekil 5.1. İki boyutlu yaklaşım örneği

Ölçüm değerlerimiz L_1 ve L_2'dir ve F fonksiyonu kullanılarak L_3 değeri bulunabilir. $X_1 = (x_1, y_1)$ ve $X_2 = (x_2, y_2)$ ise $F(X_1, X_2) = L_3$ 'ü bize verir. Daha sonra kosinüs kuralı kullanılarak α açısı bulunabilir. Fonksiyonumuz sonrasında daha küçük x değeri olan alıcı noktasını seçerek β açısını hesaplar, β açısı pozitif x yönü ile L_3 arasındaki açıdır.

Şekil 5.1'den görüleceği üzere (x_1, y_1)'den (x, y)'yi bulabilmek için x yönünde $L_1 \cos(\alpha + \beta)$ ve y yönünde $L_1 \sin(\alpha + \beta)$ eklenmelidir. Sonuç olarak (x, y) verici noktası Eş. 5.1'deki gibi bulunur.

$$(x_1 + L_1 \cos(\alpha + \beta),\ y_1 + L_1 \sin(\alpha + \beta)) \tag{5.1}$$

Tabii ki bu bize iki sonuçtan sadece birini verir. Diğer çözüm (x, y)'nin L_3'e göre yansımasıdır. Bu da Eş. 5.2 olarak hesaplanır.

$$(x_1 + L_1 \cos(\alpha - \beta),\ y_1 - L_1 \sin(\alpha - \beta)) \tag{5.2}$$

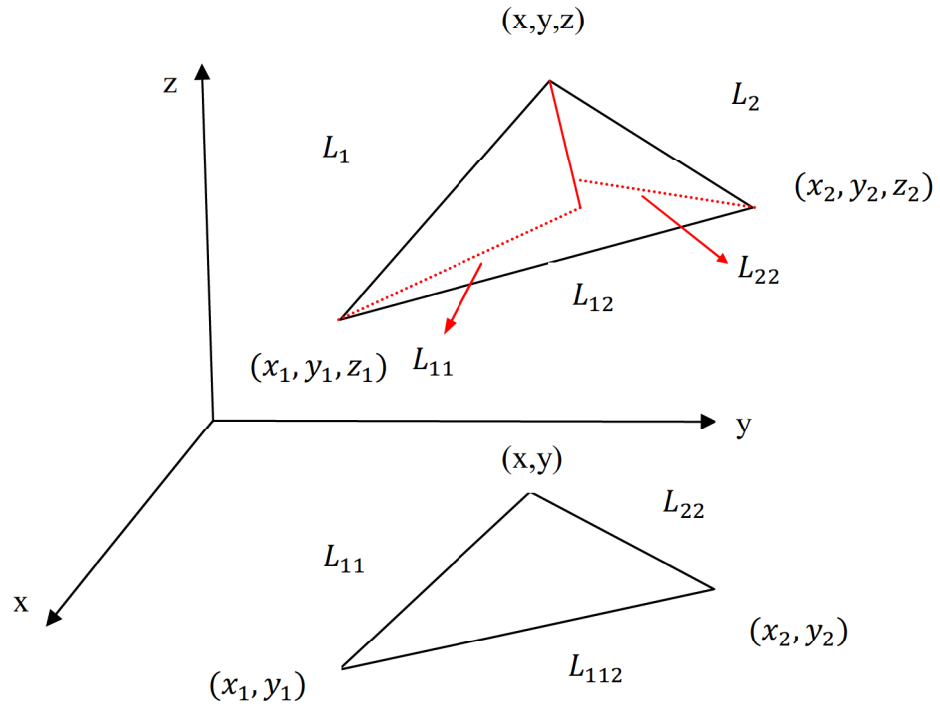
En sonunda fonksiyon bu iki çözümden hangisinin üçüncü noktaya yakın olduğunu kıyaslama yaparak belirler ve küçük olanını seçer. Locate2 fonksiyon kodu aşağıda verilmiştir.

```
function out=locate2(x1,y1,L1,x2,y2,L2,x3,y3)
   X1=[x1;y1];
   X2=[x2;y2];
   X3=[x3;y3];
   L12=F(X1,X2);

if x1<=x2

   alpha=acos((L1^2+L12^2-L2^2)/(2.*L1.*L12));
   beta=atan((y2-y1)/(x2-x1));
   x01=x1+L1.*cos(alpha+beta);
   y01=y1+L1.*sin(alpha+beta);
   x02=x1+L1.*cos(alpha-beta);
   y02=y1-L1.*sin(alpha-beta);
   X01=[x01;y01];
   X02=[x02;y02];

else

   alpha=acos((L2^2+L12^2-L1^2)/(2.*L2.*L12));
   beta=atan((y1-y2)/(x1-x2));
   x01=x2+L2.*cos(alpha+beta);
   y01=y1+L2.*sin(alpha+beta);
   x02=x2+L2.*cos(alpha-beta);
   y02=y1-L2.*sin(alpha-beta);
   X01=[x01;y01];
   X02=[x02;y02];
end
   if F(X01,X3)<F(X02,X3)
      out=[x01;y01];
   else out=[x02;y02];
end
```

5.3.2 Üç boyutlu yaklaşım (locate3)

Üç boyutlu locate3(x1,y1,z1,L1,x2,y2,z2,L2,x3,y3,X) fonksiyonu üç alıcı koordinatı ile uçak transponderinden gelen z yani yükseklik bilgisini kullanarak bulmak istediğimiz üç boyutlu noktanın koordinatlarını hesaplayan fonksiyondur.

Bu fonksiyonda alıcıların ve uçağın z koordinat bilgileri kullanılarak ilgili noktalar iki boyuta indirgenir (Şekil 5.2), mevcut z noktasına ek olarak x ve y noktaları hesaplanarak üç boyutlu nokta bulunabilir.



Şekil 5.2. Üç boyutlu yaklaşım örneği

Şekil 5.2'den görüleceği üzere alttaki üçgendeki L_{11} ve L_{22} uzunluklarını bulabilmek için Pisagor bağlantıları kullanabilir.

$$L_{11} = \sqrt{(L_1)^2 - (z - z_1)^2} \tag{5.3}$$

$$L_{22} = \sqrt{(L_2)^2 - (z - z_2)^2} \tag{5.4}$$

L_{112} uzunluğu da elimizde var olan koordinatlardan, F fonksiyonu kullanılarak hesaplanabilir. Dolayısıyla üç boyutlu fonksiyon iki boyuta indirgenmiş olur. Bu noktadan sonra artık iki boyutlu çalışmada anlatılan kurallar uygulanarak x ve y noktaları bulunabilir, z noktası zaten mevcuttu, dolayısıyla istenilen koordinatlar elde edilmiş olunur.

Locate3 fonksiyonunun kodu aşağıdaki gibidir:

```
function out=locate3(x1,y1,z1,L1,x2,y2,z2,L2,x3,y3,X)

   L11 = sqrt(L1^2 - (X(3)-z1)^2);
   L22 = sqrt(L2^2 - (X(3)-z2)^2);

   b=locate2(x1,y1,L11,x2,y2,L22,x3,y3);
   b(3)=X(3);
   out= b;

end
```

5.4. Simülasyon

Simülasyon çalışmaları da iki boyutlu ve üç boyutlu olarak gerçekleştirilmiştir. İki boyutlu olan simulation2 programında, öncelikle kullanıcıdan sırasıyla alıcıların (x, y) koordinatlarını girmesi istenmektedir. İlgili koordinatlar girildikten sonra program kullanıcıdan, her bir ölçüm için yüzdelik hataları girmesini ister. Çünkü gerçek hayatta ölçümlerde mutlaka bir hata payının olduğu programımızda göz ardı edilmemiştir.

Daha sonra program "gerçek" pozisyon üretir; bu pozisyon bizim iterasyonlarla yaklaşım yapmaya çalıştığımız pozisyondur. Sonrasındaysa program her bir alıcı noktasından bu "gerçek" pozisyona olan gerçek uzaklıkları hesaplar. Rastgele sayı

üreticisi kullanılarak ve her bir ölçüm için yüzdelik hatalara ve gerçek pozisyonlara bu sayılar dağıtılarak, rastgele yalancı ölçümler elde edilir.

Bunları yaptıktan sonra program locate2 fonksiyonunu kullanarak gerçek pozisyona bir yaklaşım yapar. Bu arada üretilen bu rastgele pozisyonla girilmiş alıcı pozisyonları arasında tutarsızlık oluşuyorsa (denklemlerden herhangi biri sıfır veya sonsuz oluyorsa, ya da oluşturulan hiperbol veya hiperboloidlerden herhangi birisi diğerleriyle kesişmiyorsa vb.) istenilen grafik çizdirilemeyecektir. Bu nedenle program kullanıcıdan, yeni bir pozisyon üretmesi için sıfır girmesini ister. Böylece tutarlı pozisyon üretilebilmekte ve geçerli yaklaşımlar yapılabilmektedir.

Bu ilk yaklaşım değerini kullandıktan sonra program, en küçük kareler yöntemini kullanarak yeni pozisyon üretir. Bu yapılırken birçok iterasyon gerçekleştirilir ve ilgili iterasyon vektörü 0.001'den küçük olana kadar iterasyonlara devam edilebilir. İstenilirse bu sayı daha sonra ihtiyaca göre değiştirilebilir. Programa böyle bir sayı konulmasının sebebi iterasyon döngülerinden dolayı programın yavaşlayıp, durmasını önlemektir.

Her bir üretilen pozisyondan (yaklaşık pozisyon-iterasyon) sonra, kullanıcı isterse direkt olarak son cevaba atlayabilir. Bunun için, programda sorulan soruya 0 girmesi yeterlidir. Ya da kullanıcı 1 girerek sonuca nasıl yaklaşım yapıldığını aşama aşama ilerleyerek görebilir. Bütün bu yapılan iterasyonlar ve bulunan pozisyonlar da grafik çizdirilerek desteklenmektedir.

Gerçek pozisyon hiperbol ya da hiperboloidlerin kesişim noktasında yıldız şeklinde çizdirilmektedir. Aşama aşama ilerlerken elde edilen sonuç aynı grafikte mavi çarpı olarak, en son sonuç ise kırmızı çarpı olarak çizdirilmektedir.

Simulation2 programı için kod aşağıdaki gibidir:

```
a=0.001; %iterasyonların durması için gerekli değer
prompt={'x1','y1','x2','y2','x3','y3'};
```

```
name='Alıcı koordinatları:'; % alıcı koordinatlarının seçimi
numlines=1;
defaultanswer={'0','0','0','0','0','0'};
answer0=inputdlg(prompt,name,numlines,defaultanswer);

x1=str2num(answer0{1});
y1=str2num(answer0{2});
x2=str2num(answer0{3});
y2=str2num(answer0{4});
x3=str2num(answer0{5});
y3=str2num(answer0{6});

X1=[x1;y1]; %alıcı koordinatlarının vektöre çevirimi
X2=[x2;y2];
X3=[x3;y3];

i=0;
prompt={'Error 1','Error 2', 'Error 3'};
name='Ölçüm Hataları:'; % muhtemel ölçüm hataları
numlines=1;
defaultanswer={'1','1','1'};
answer2=inputdlg(prompt,name,numlines,defaultanswer);

for i=0:100
rand('seed',i);
randn('seed',0);

error1=str2num(answer2{1}).*0.01;
error2=str2num(answer2{2}).*0.01;
error3=str2num(answer2{3}).*0.01;
D=[error1^2;error2^2;error3^2]; %Bu satırlar P ağırlık matrisini,
P=diag(D);     % oluşturur, diagonal elemanları hatalar olan diagonal matris
```

```
X=10.*rand(2,1); %rastgele 'gerçek' pozisyon
D1=F(X,X1); %herbir alıcının 'gerçek' pozisyona mesafesi
D2=F(X,X2);
D3=F(X,X3);

iterasyon_sayisi=0;

L1=D1+D1.*randn.*error1; %rastgele hatalara gerçek pozisyonu ekleyerek yapılan
ölçümler
L2=D2+D2.*randn.*error2;
L3=D3+D3.*randn.*error3;

X0=locate2(x1,y1,L1,x2,y2,L2,x3,y3); % Girilenlere göre bulunan nokta

hyperbolplot (x1,x2,x3,y1,y2,y3,X)
hold on;

options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt={'Kesişimler yetersizse, yeni sayı üretmek için 0 a tuşlayın. Devam etmek
için 1 e basın'};
name='Kesişimler:';
numlines=1;
defaultanswer={'0'};
examine=inputdlg(prompt,name,numlines,defaultanswer,options);
%examine=questdlg('Kesişimler yetersizse yeni sayı üretmek
istermisiniz?','Input','evet','hayır');
%Bazı denklemler sıfıra ya da sonsuza gittiğinde yeterince kesişim elde
%edilemez bu yüzden kullanıcıya yeni sayı üretmesi için fırsat tanır.

if(str2num(examine{1})==0)
```

```
i=i+1;
else
break
end
end
b=1;
moddX=a+1; % a'dan buyuk mod(dX) degeri atıyor

options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt={'İç iterasyonlara bakmak istermisiniz? Evet için 1 yazın, yoksa 0 olarak bırakın'};
name='Iterasyonlar:';
numlines=1;
defaultanswer={'0'};
examine=inputdlg(prompt,name,numlines,defaultanswer,options);
%examine=questdlg('İc iterasyonlara bakmak istermisiniz?','Input','evet','hayır');
%Herbir ardışık hesaplanmış pozisyona bakabilmek için kullanıcıya fırsat tanır.

while (moddX>=a)
W=([F(X0,X1) F(X0,X2) F(X0,X3)]-[L1 L2 L3])'; % Örtüm vektorü
A=[((X0-X1)/F(X0,X1))';((X0-X2)/F(X0,X2))';((X0-X3)/F(X0,X3))'];%Tasarım matrisi
dX=-1*inv(A'*P*A)*A'*P*W; % Standart sonuca göre Xo sarsımını (perturbasyon) hesaplar
moddX=sqrt(dX'*dX); %Sarsım (perturbasyon) ölçüsü

X0=X0+dX; % pozisyon noktasının yeni değeri
iterasyon_sayisi=iterasyon_sayisi+1;%Kaç iterasyon olduğunu sayıyor

if(str2num(examine{1})==1)&&(b~=0)
```

```
    plot(X0(1,1),X0(2,1),'x','MarkerSize',12,'MarkerEdgeColor','b')
    hold on
    options.Resize='on';

options.WindowStyle='normal';
options.Interpreter='tex';
prompt={'Sonucu görmek için 1 yazın. Bir sonraki iterasyon için 0 yazın'};
name='Proceed?:';
numlines=1;
defaultanswer={'0'};

moveon=inputdlg(prompt,name,numlines,defaultanswer,options);
% moveon=questdlg('Sonucu görmek için 1 yazın?','Input','plot next','Jump to final
value','plot next');
%Eğer kullanıcı iç iterasyonlara bakmayı seçerse son değere atlayabilir value

if (str2num(moveon{1})==1)
b=0;
end
 if (iterasyon_sayisi>1000) % programın çakılmaması için iterasyonları sınırlıyor
    moddX=a-1;
end
end
end %X0 daki değişimler a'dan düşük olmadıkça bu süreç devam edecek

X
hesaplanan=X0+dX
iterasyon_sayisi

plot(hesaplanan(1,1),hesaplanan(2,1),'x','MarkerSize',12,'MarkerEdgeColor','m')
hold;
```

Simulation3 programı ise üç boyutlu çalışma için hazırlanan programdır. Bu programda simulation2'ye göre ufak tefek farklılıklar vardır.

Öncelikle söz konusu üç boyutlu simülasyon için hyperboloidplot ve locate3 fonksiyonları hazırlanmıştır. Çizdirim ve yer kestirim için program anılan fonksiyonları kullanmaktadır. Sonrasında simulation3 programında alıcı anten sayısı üçten dörde çıkartılmıştır. Bu yüzden kullanıcıdan dört adet x,y,z koordinatı girmesi istenir. Ayrıca dört adet alıcı istenildiği için dört adet de hata oranı girilmesi istenir. Bu programda unutulmaması gereken diğer bir konu da uçaktan elde edilen z bilgisinin kullanılmasıdır. Diğer her şey, iki boyutlu simulation2 programındaki gibi işlemektedir.

Simulation3 programının kodu aşağıdaki gibidir:

```
a=0.001; %iterasyonların durması için gerekli değer
name='Alıcı koordinatları:'; %4 alıcı koordinatlarının seçimi
numlines=1;
prompt={'x1','y1','z1','x2','y2','z2','x3','y3','z3','x4','y4','z4'};
defaultanswer={'0','0','0','0','0','0','0','0','0','0','0','0'};
numlines=1;
answer0=inputdlg(prompt,name,numlines,defaultanswer);

x1=str2num(answer0{1});
y1=str2num(answer0{2});
z1=str2num(answer0{3});
x2=str2num(answer0{4});
y2=str2num(answer0{5});
z2=str2num(answer0{6});
x3=str2num(answer0{7});
y3=str2num(answer0{8});
z3=str2num(answer0{9});
x4=str2num(answer0{10});
```

```
y4=str2num(answer0{11});
z4=str2num(answer0{12});

X1=[x1;y1;z1]; %alıcı koordinatlarının vektöre çevirimi
X2=[x2;y2;z2];
X3=[x3;y3;z3];
X4=[x4;y4;z4];
 i=0;
 prompt={'Error 1','Error 2', 'Error 3', 'Error 4'};
name='Ölçüm Hataları:'; % muhtemel ölçüm hataları
numlines=1;
defaultanswer={'1','1','1','1'};
answer2=inputdlg(prompt,name,numlines,defaultanswer);

for i=1:100
rand('seed',i);
randn('seed',0);

error1=str2num(answer2{1}).*0.01;
error2=str2num(answer2{2}).*0.01;
error3=str2num(answer2{3}).*0.01;
error4=str2num(answer2{4}).*0.01;

D=[error1^2;error2^2;error3^2;error4^2]; %Bu satırlar P ağırlık matrisini,
P=diag(D);   % oluşturur, diagonal elemanları hatalar olan diagonal matris

X=10.*rand(3,1);%rastgele 'gerçek' pozisyon

D1=F(X,X1); %herbir alıcının 'gerçek' pozisyona mesafesi
D2=F(X,X2);
D3=F(X,X3);
D4=F(X,X4);
```

```
iterasyon_sayisi=0;

L1=D1+D1.*randn.*error1; %rastgele hatalara gerçek pozisyonu ekleyerek yapılan
ölçümler
L2=D2+D2.*randn.*error2;
L3=D3+D3.*randn.*error3;
L4=D4+D4.*randn.*error4;

X0=locate3(x1,y1,z1,L1,x2,y2,z2,L2,x3,y3,X); % Girilenlere göre bulunan nokta

hyperboloidplot (x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,X)
hold on;

options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt={'Kesişimler yetersizse, yeni sayı üretmek için 0 a tuşlayın. Devam etmek
için 1 e basın'};
name='Kesişimler:';
numlines=1;
defaultanswer={'0'};
examine=inputdlg(prompt,name,numlines,defaultanswer,options);
%examine=questdlg('Kesişimler yetersizse yeni sayı üretmek
istermisiniz?','Input','evet','hayır');

%Bazı denklemler sıfıra ya da sonsuza gittiğinde yeterince kesişim elde
%edilemez bu yüzden kullanıcıya yeni sayı üretmesi için fırsat
%tanır

if(str2num(examine{1})==0)
i=i+1;
else
```

```
break
end
end

b=1;
moddX=a+1; % a'dan buyuk mod(dX) degeri atıyor

options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt={'İç iterasyonlara bakmak istermisiniz? Evet için 1 yazın, yoksa 0 olarak
bırakın'};
name='Iterasyonlar:';
numlines=1;
defaultanswer={'0'};
examine=inputdlg(prompt,name,numlines,defaultanswer,options);
%examine=questdlg('İc iterasyonlara bakmak istermisiniz?','Input','evet','hayır');
%Herbir ardışık hesaplanmış pozisyona bakabilmek için kullanıcıya fırsat
%tanır

while (moddX>=a)
W=([F(X0,X1) F(X0,X2) F(X0,X3) F(X0,X4)]-[L1 L2 L3 L4])'; %Örtüm vektorü
A=[((X0-X1)/F(X0,X1))';((X0-X2)/F(X0,X2))';((X0-X3)/F(X0,X3))';((X0-
X4)/F(X0,X4))']; %Tasarım matrisi
dX=-1*inv(A'*P*A)*A'*P*W; % Standart sonuca göre Xo sarsımını (perturbasyon)
hesaplar
moddX=sqrt(dX'*dX); %Sarsım (perturbasyon) ölçüsü

X0=X0+dX; % pozisyon noktasının yeni değeri
iterasyon_sayisi=iterasyon_sayisi+1;%Kaç iterasyon olduğunu sayıyor
```

```
if(str2num(examine{1})==1)&&(b~=0)
plot3(X0(1,1),X0(2,1),X0(3,1),'x','MarkerSize',12,'MarkerEdgeColor','b')
   hold on
   options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt={'Sonucu görmek için 1 yazın. Bir sonraki iterasyon için 0 yazın'};
name='Proceed?:';
numlines=1;
defaultanswer={'0'};

moveon=inputdlg(prompt,name,numlines,defaultanswer,options);
%moveon=questdlg('Sonucu görmek için 1 yazın?','Input','plot next','Jump to final
value','plot next');
%Eğer kullanıcı iç iterasyonlara bakmayı seçerse son değere atlayabilir value

if (str2num(moveon{1})==1)
b=0;
end

if (iterasyon_sayisi>1000) % programın durmamasi için iterasyonları sınırlıyor
   moddX=a-1;
end
end
end %X0 daki değişimler a'dan düşük olmadıkça bu süreç devam edecek

 X
hesaplanan=X0+dX
iterasyon_sayisi
plot3(hesaplanan(1,1),hesaplanan(2,1),hesaplanan
(3,1),'x','MarkerSize',12,'MarkerEdgeColor','m')
hold;
```

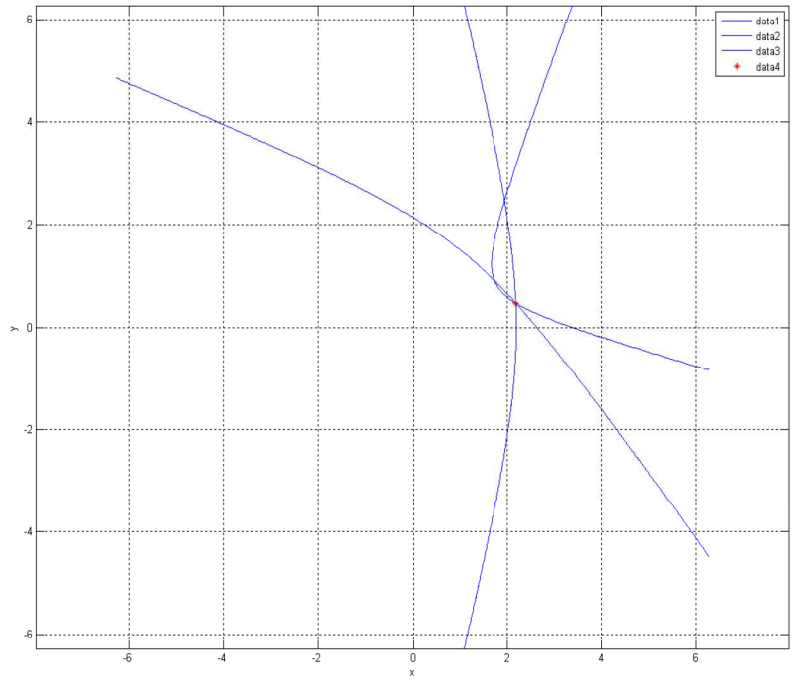
6. SİMÜLASYON ÇIKTIRLARI VE DEĞERLENDİRMELERİ

Bu bölümde, yapmış olduğumuz simülasyon programından elde ettiğimiz çıktılar anlatılacaktır.

6.1. İki Boyutlu Simülasyon (Simulation2)

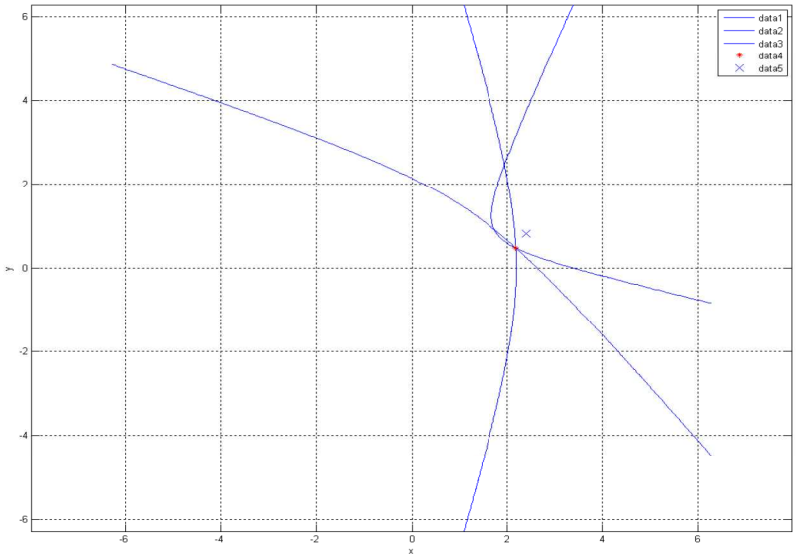
İki boyutlu simulation2 programımızı komut satırına simulation2 yazarak çalıştırdığımızda program bizden alıcı koordinatlarını girmemizi ister.

Üç alıcının koordinatlarını programımıza sırasıyla (2,1), (6,0), (3,4) olarak giriyoruz. Alıcı koordinatlarını girdikten hemen sonra bizden yüzdelik hataları girmemiz istenir. Söz konusu hataları da öncelikle varsayılan değerler olan 1 olarak giriyoruz. İlgili hataları da girdikten sonra program bize görmek istediğimiz kesişimleri elde edip etmediğimizi sorar. Şekil 6.1'den de görüleceği gibi kesişimler gayet net olarak elde edilmiş ve uçak pozisyonu kırmızı renkte "*" sembolüyle çizdirilmiştir.



Şekil 6.1. Uçak pozisyonu ve hiperbollerin kesişimi

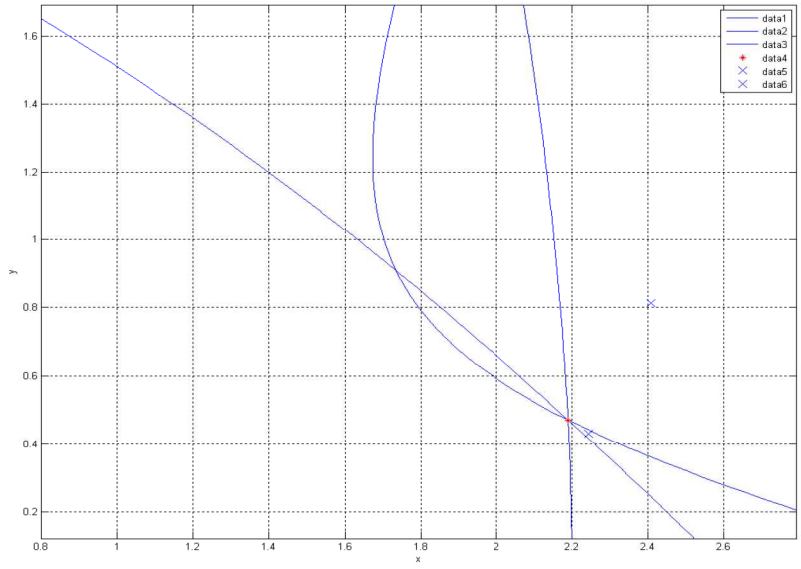
Kesişimler istediğimiz gibi elde edildiği için 1'e basarak programımıza devam ediyoruz. Bu aşamada program bize iç iterasyonlara bakmak isteyip istemediğimizi sorar. Biz de iterasyonları görmek istediğimiz için 1'e basıp programa devam ederek Şekil 6.2'deki ilk iterasyon sonucunu elde ediyoruz. Burada program ilk iterasyon için hesaplayarak elde ettiği pozisyon bilgisini mavi renkte "x" işaretiyle çizdirmektedir.



Şekil 6.2. İlk iterasyon sonucu

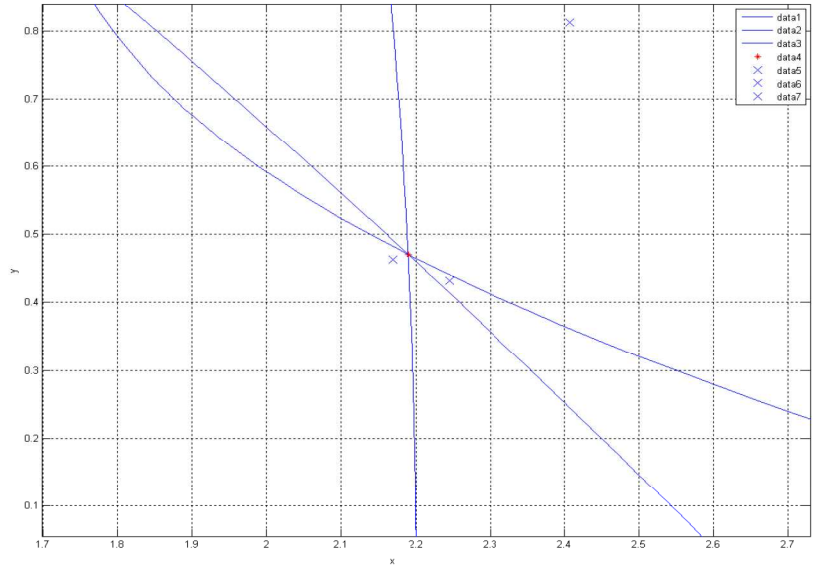
Bu aşamada istersek 1 tuşlayarak, sırasıyla bütün iterasyonlara bakabiliriz, ya da 0 tuşlayıp direkt en son sonuca gidebiliriz.

Biz üretilen pozisyonları ve gerçek pozisyona nasıl yaklaşıldığını görmek istediğimiz için iterasyonlar bitene kadar 0 tuşlayıp devam ediyoruz. Şekil 6.3'te ikinci iterasyon gösterilmiştir.



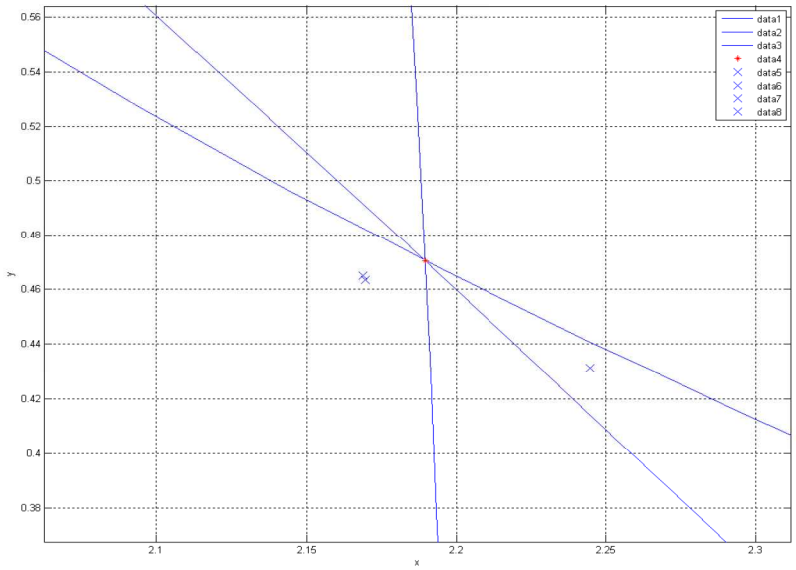
Şekil 6.3. İkinci iterasyon sonucu

Şekil 6.4'te üçüncü iterasyon sonucu elde edilen grafik gösterilmiştir.



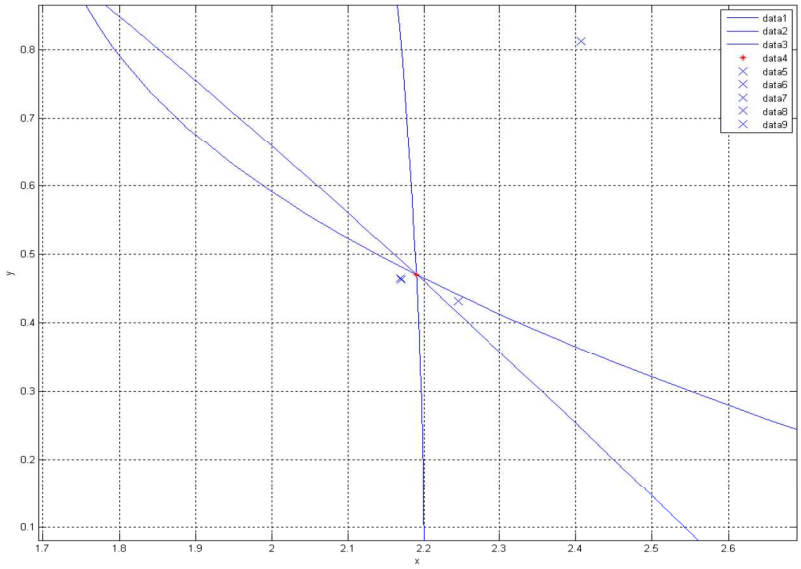
Şekil 6.4. Üçüncü iterasyon sonucu

Şekil 6.5'te ise dördüncü iterasyon sonucu elde edilen grafik görülmektedir.



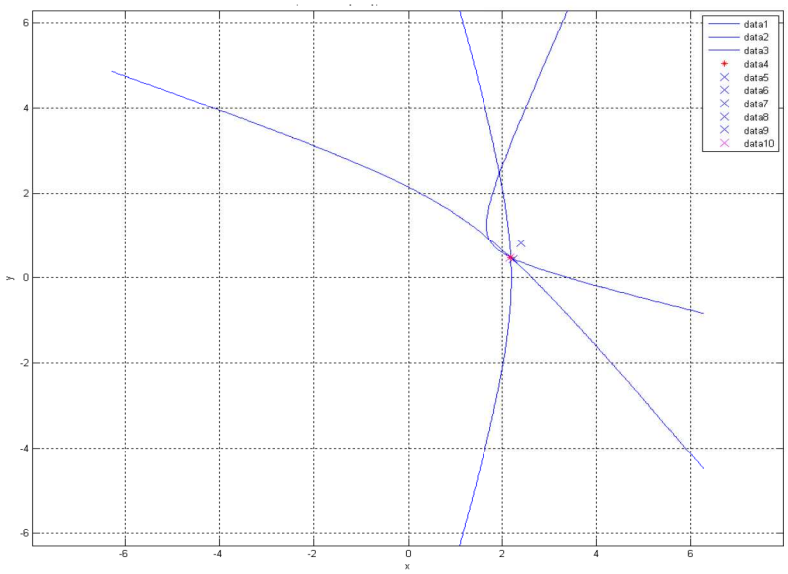
Şekil 6.5. Dördüncü İterasyon

Şekil 6.6'da de beşinci ve son iterasyon sonucu elde edilen sonuç görülmektedir.



Şekil 6.6. Beşinci iterasyon sonucu

Sonuç olarak, Şekil 6.7'deki gibi program tarafından hesaplanan uçak pozisyonunu elde ediyoruz (kırmızı çarpı işareti).

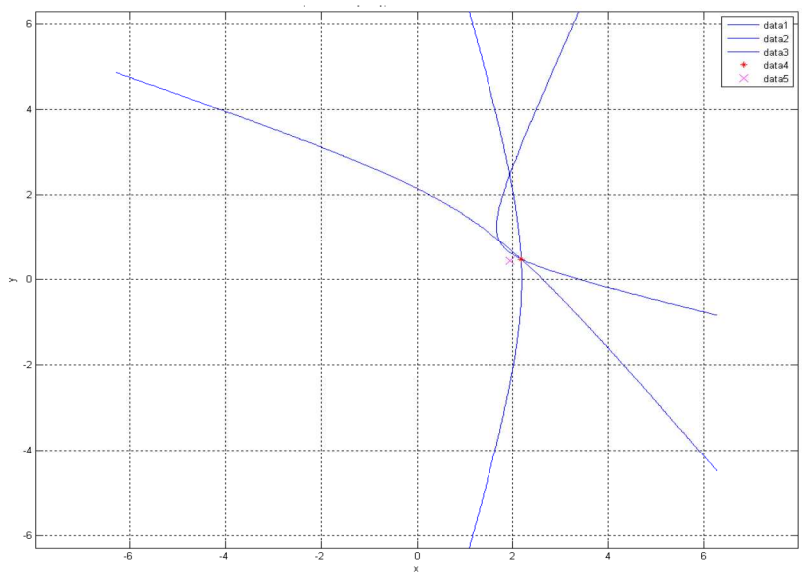


Şekil 6.7. Sonuçta elde edilen pozisyon gösterimi

Ayrıca uçağın gerçek pozisyonu, program tarafından hesaplanan pozisyon ve iterasyon sayısı da Matlab ana penceresinde yazdırılmaktadır.

Bu örneğimizde gerçek pozisyon (2,1896; 0,4704) olarak üretilmiş, girilen hatalar ile bizim yaptığımız yaklaşımlar sonucunda (2,1688;0,4650) olarak 5 iterasyon sonucunda hesaplanmıştır. Görüldüğü gibi hata oranları düşük tutulduğu için, çok yakın bir değer elde edilebilmiştir.

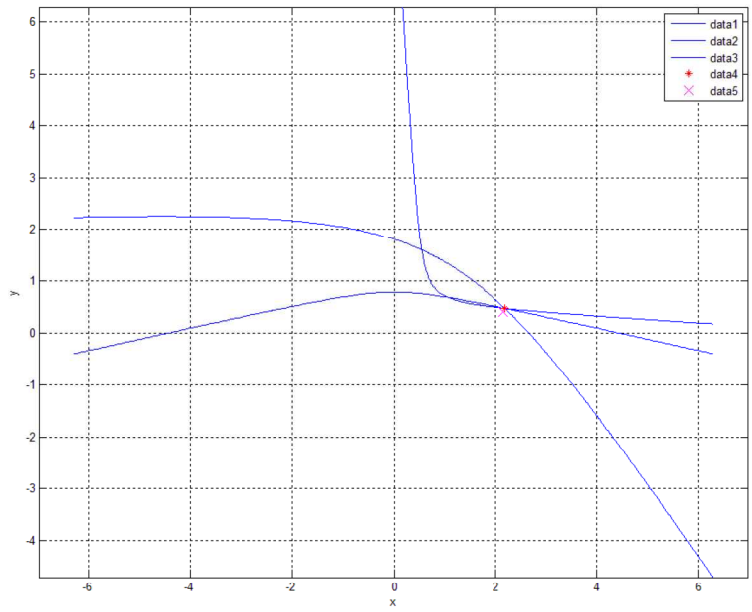
Aynı örnekte hatalarımızı %10 seviyelerine çekerek artırır ve yaklaşım yaparsak Şekil 6.8'i elde ederiz.



Şekil 6.8. %10'luk hata oranları sonucu elde edilen pozisyon grafiği

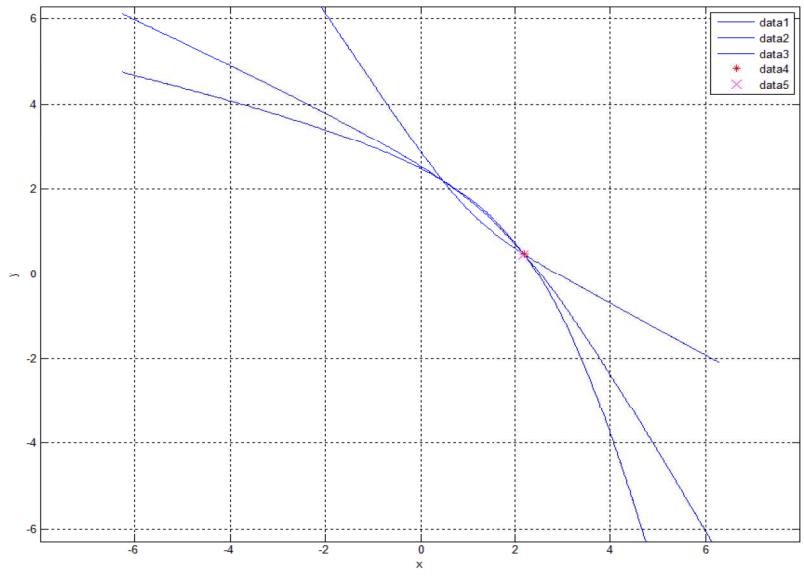
Görüldüğü gibi ilkine nazaran daha hatalı sonuçlar elde edilmiştir. (2,1896; 0,4704) gerçek değerine karşılık program tarafından gerçekleştirilen yedi iterasyondan sonra elde edilen sonuç (1,9567; 0,4369)'dur. Hata oranları arttıkça uçak pozisyonundan giderek uzaklaşılmaktadır.

İkinci örneğimizde alıcı koordinatlarını sırasıyla (3,6), (0,2) ve (1,1) olarak hata oranlarını da varsayılan (1,1,1) olarak berlileyerek programımızı çalıştırdığımızda bu sefer altı iterasyon sonucunda elde edilen sonuç (2,1631,0;4082) olmuştur. Şekil 6.9'da elde edilen sonuç grafiği gösterilmiştir.



Şekil 6.9. İki boyutlu ikinci örnek grafiği

Üçüncü örneğimizde alıcı koordinatlarını (2,2), (6,6), (4,4) olarak, hata oranlarını yine varsayılan değerler olan (1,1,1) olarak giriyoruz. Bu sefer iki iterasyon neticesinde elde ettiğimiz sonuç (2,1696, 0,4593) olmaktadır. Şekil 6.10'da üçüncü örneğimiz sonucunda elde ettiğimiz grafik gösterilmiştir.

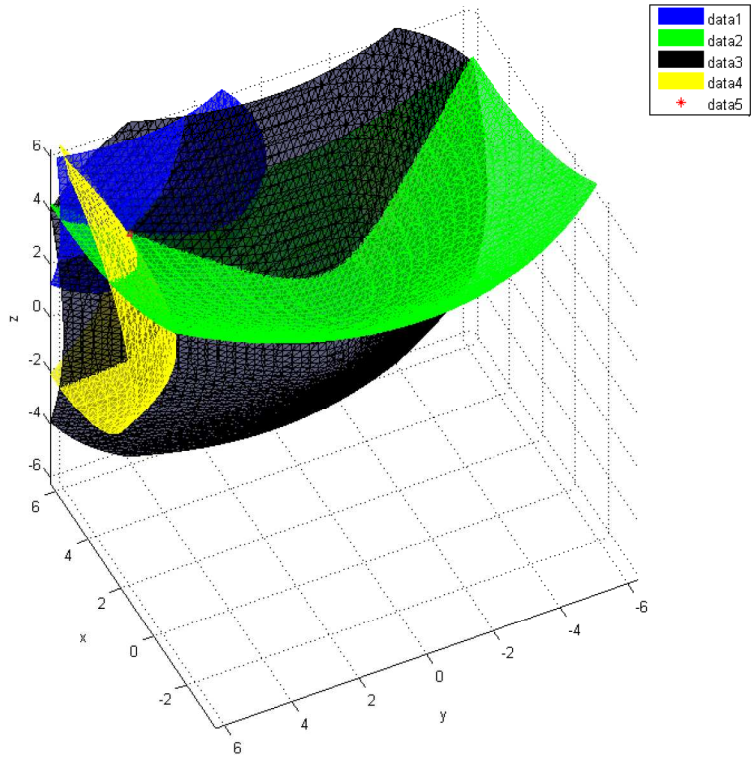


Şekil 6.10. İki boyutlu üçüncü örnek grafiği

6.2. Üç Boyutlu Simülasyon (Simulation3)

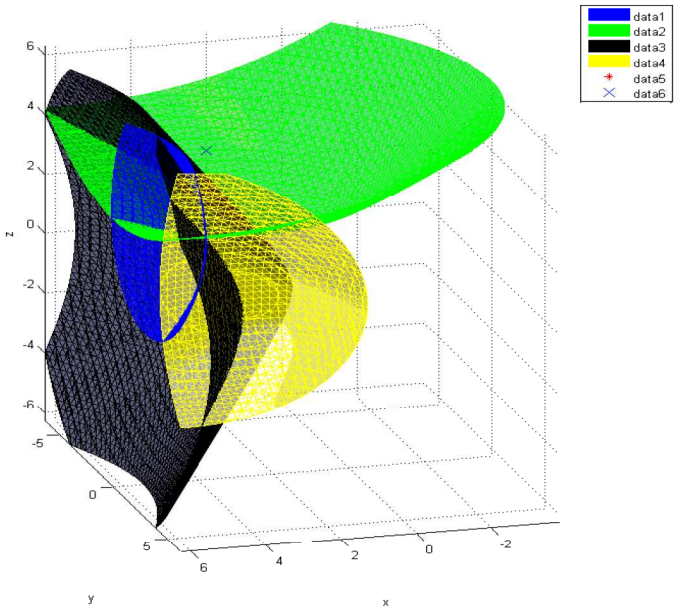
Komut satırına simulation3 yazdığımızda bizden alıcı koordinatlarını (z koordinatı dahil) üç boyutlu olarak girmemiz istenir. Bu programda iki boyutludan farklı olarak 3 yerine 4 tane alıcı koordinatı belirlenmiştir. Dolayısıyla bizden, 4 adet x, y ve z noktası ile 4 adet hata oranı girmemiz istenir.

Alıcı koordinatlarımızı sırasıyla (6,2,2), (1,0,3), (2,0,0) ve (3,5,1) olarak giriyoruz, hata oranlarını ise gene varsayılan değerler (1,1,1,1) olarak bırakıyoruz. Bu değerler sonucunda ilk üretilen gerçek pozisyon Şekil 6.11'deki gibidir.



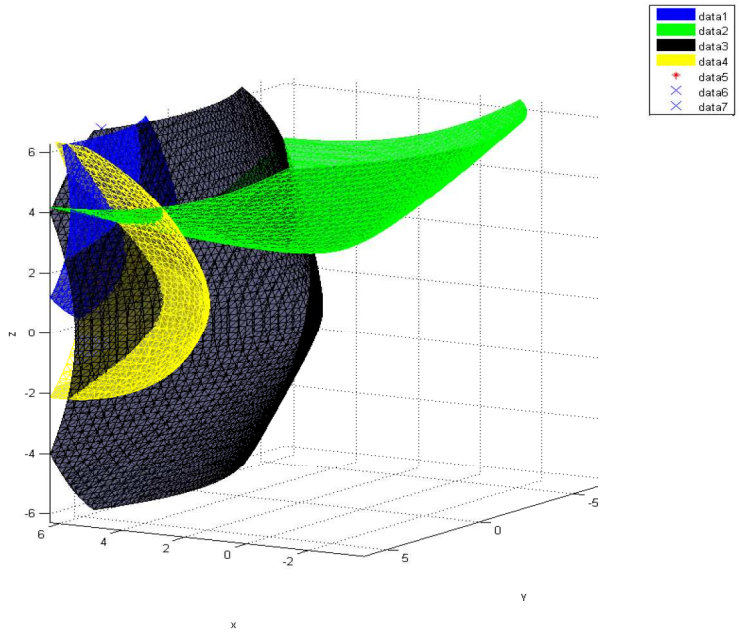
Şekil 6.11. Uçak pozisyonu ve hiperboloidlerin kesişimi

Daha sonra istediğimiz kesişimler elde edilebildiği için 1'e basarak devam ediyoruz ve iterasyonlara geçiyoruz. Karşımıza gelen iç iterasyonlara bakmak istermisiniz sorusuna da 1'e basıp evet diyoruz ve ilk iterasyon sonucu olarak Şekil 6.12'yi elde ediyoruz.



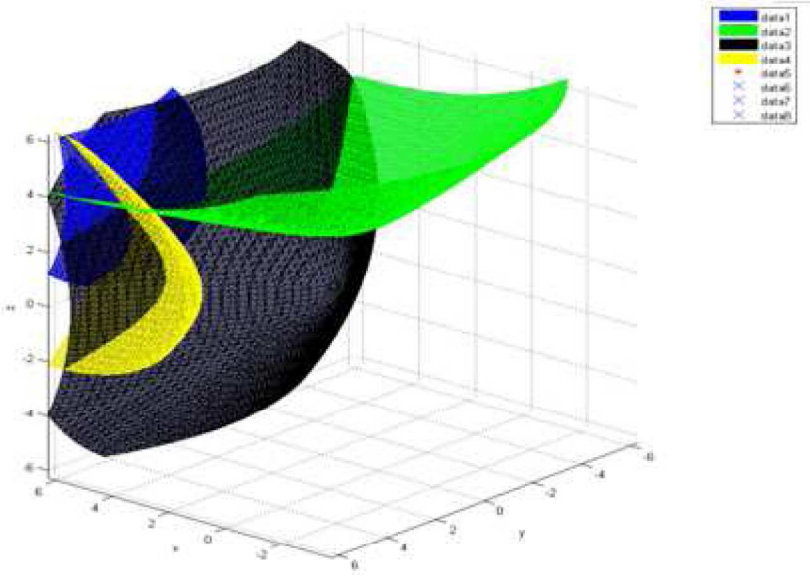
Şekil 6.12. İlk iterasyon sonucu

Sonrasında 0'a basarak iterasyonlarımıza devam ediyoruz. Şekil 6.13'te ikinci iterasyon sonucu elde edilen grafik gösterilmiştir.



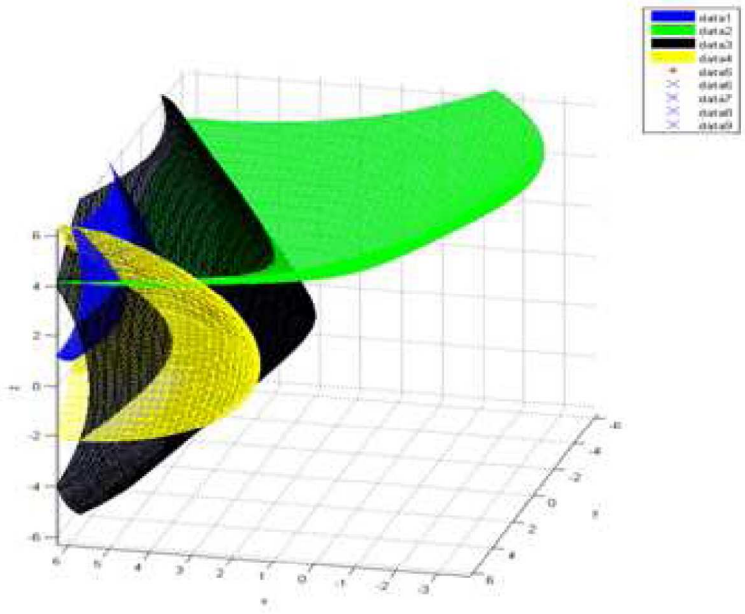
Şekil 6.13. İkinci iterasyon sonucu

Şekil 6.14'de üçüncü iterasyon sonucu görülmektedir. Sonuca gittikçe yaklaşmaktayız.



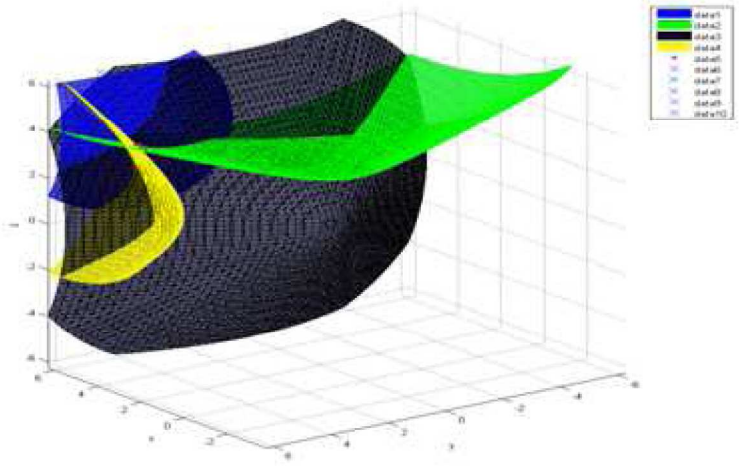
Şekil 6.14. Üçüncü iterasyon sonucu

Şekil 6.15'te dördücü iterasyon sonucu görülmektedir.



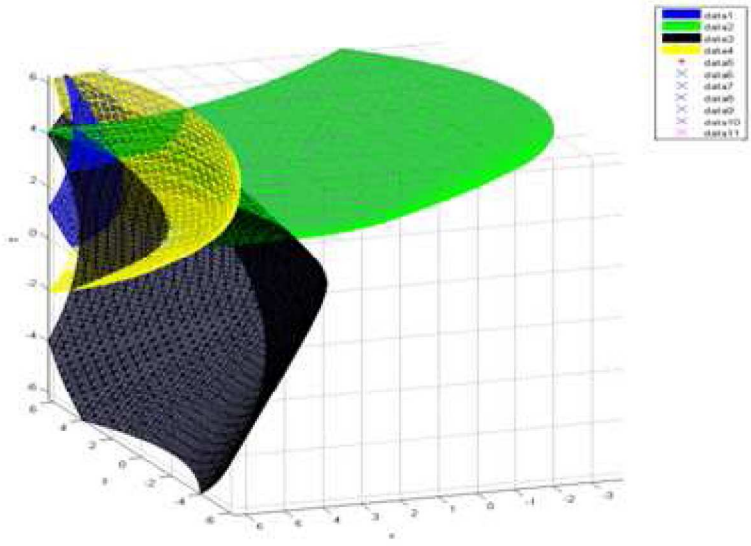
Şekil 6.15. Dördüncü iterasyon sonucu

Şekil 6.16'da beşinci ve son iterasyon sonucu görülmektedir.



Şekil 6.16. Beşinci iterasyon sonucu

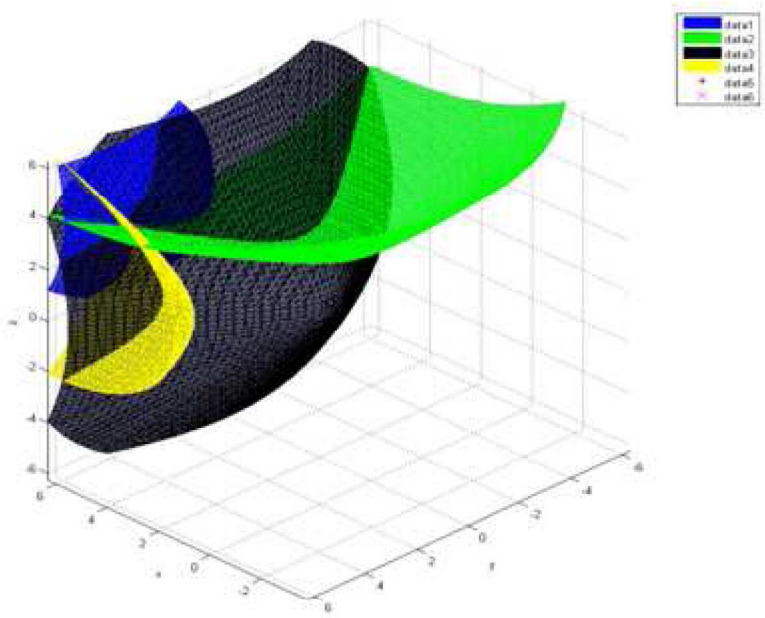
Sonuçta hesaplanan pozisyon ise Şekil 6.17'de gösterilmiştir. Bu bölümde elde edilen bütün grafiklerin sağ üst köşesinde veri alanı yer almaktadır. Bu alanda grafikte yer alan bilgiler açıkça görülmektedir. Şekilde net olarak görülmese de bu alandan grafiğimizde var olan verileri görebilmekteyiz.



Şekil 6.17. Sonuçta elde edilen pozisyon gösterimi

Matlab ana penceresine döndüğümüzdeyse sayısal sonuçları görebilmekteyiz. Buna göre (5,1291; 4,6048; 3,5284) olan gerçek pozisyon 5 adet iterayson sonucunda (5,1344; 4,6363; 3,5083) olarak elde edilmiştir.

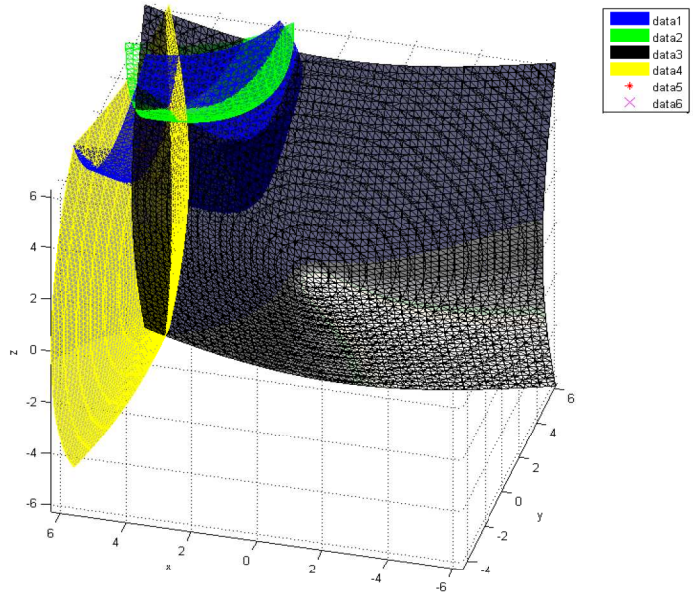
Aynı problemi hatalarımızı sırasıyla (10,15,5,10)'a çekerek yaparsak Şekil 6.16'yı elde ederiz.



Şekil 6.18. % (10,15,5,10)'luk hata oranları sonucu elde edilen pozisyon

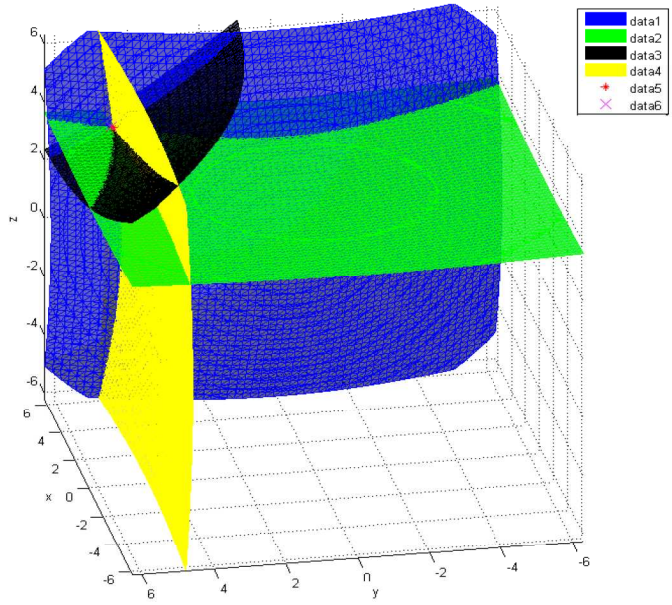
Görüldüğü gibi ilgili hata oranlarını girdiğimizde 6 iterasyon neticesinde (5,4193; 5,1155; 3,3652) sonucu elde edilmiştir. Yani hata oranlarını arttırdıkça gerçek değere olan yaklaşımımız gittikçe uzaklaşmaktadır. Bu da programdan istediğimiz sonuçları elde edebildiğimizin bir göstergesidir.

Üç boyutlu ikinci örneğimizde alıcı koordinatlarımızı sırasıyla (2,1,3), (4,6,6), (0,2,0), (8,1,0) olarak ve hata oranlarımızı da varsayılan değerler olan (1,1,1,1) olarak giriyoruz. Girdiğimiz bu veriler neticesinde yedi iterasyon sonucunda pozisyon (5,1515; 4,6480; 3,4765) olarak elde edilmiştir. Şekil 6.19'da sonuç grafiği gösterilmiştir.



Şekil 6.19. Üç boyutlu ikinci örnek sonuç grafiği

Üçüncü örneğimizde ise alıcı koordinatlarımızı sırasıyla (4,0,0), (0,0,6), (2,6,8), (0,8,0) olarak hata oranlarını da (1,1,1,1) olarak giriyoruz. İki iterasyon neticesinde pozisyon (5,1771; 4,6500; 3,5300) olarak elde edilmiştir. Şekil 6.20'de elde edilen sonuç grafiği gösterilmiştir.



6.20. Üç boyutlu üçüncü örnek sonuç grafiği

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada son yıllarda havacılık sektöründe, özellikle gözetim amaçlı kullanılmaya başlayan ve ileride mevcut radarların yerini alması muhtemel olan MLAT sistemler incelenmiş, TDOA yöntemiyle uçak pozisyonu çizdirilmiş ve en küçük kareler metoduyla da iterasyonlarla söz konusu pozisyona yaklaşım yapılarak pozisyon kestirilmeye çalışılmıştır.

Simülasyonların sonunda uçak pozisyonu grafiksel olarak gösterilmiş, ayrıca aşama aşama yaklaşımlar çizdirilmiştir. Bununla birlikte sayısal olarak da gerçek uçak pozisyonu ve hesaplanan uçak pozisyonu yapılan iterasyon sayısıyla birlikte çıktı olarak elde edilmiştir.

Çalışmada, göz ardı etmememiz gereken önemli verilerden birisi ölçüm hatalarıdır. Çünkü gerçek hayatta yapılan ölçümler, hiçbir zaman hatasız değildir. Dolayısıyla bu çalışmadaki simülasyonlarda kullanıcının kendi girebileceği yüzdelik hata oranları tanımlanmış ve bunlar rastgele olarak verilere dağıtılmıştır.

Hata oranları artırılarak kıyaslama yapılmıştır ve hata oranları arttıkça gerçek pozisyondan gittikçe uzaklaşıldığı görülmüştür. Bu da hata dağılımlarının doğru yapıldığının bir göstergesidir.

Bunula birliklte farklı alıcı koordinatları belirlenerek programlar çalıştırılmış, elde edilen sonuçlar hem sayısal hem de grafiksel olarak gösterilmiştir. İlgili örneklerde hata oranları sabit tutulmuş ve gerçek pozisyona yakın sonuçların elde edildiği görülmüştür.

Bu çalışmada kullanılan yaklaşım yöntemi en küçük kareler yöntemidir. Aynı şekilde uzatılmış kalman filtresi, bias regresyon yöntemi gibi yöntemler kullanılarak pozisyon bulma çalışmaları yapılabilir.

Dünyada birçok havaalanında/geniş alanlarda, kullanımı özellikle az maliyet oranları ve yüksek doğruluk değerlerinden dolayı yaygınlaşan MLAT sistemler üzerine yapılan çalışmalar da her geçen gün artmaktadır. Halihazırda dünyada birçok havaalanında A-SMGCS sistemlerinin bir parçası olarak kullanılmakta olan bu sistemler, artık geniş alanlarda (WAM) da yaygın olarak kullanılmaya başlayacaktır.

Bu konuyla ilgili EUROCONTROL (Avrupa Hava Seyrüseferi Emniyeti Teşkilatı; The European Organization for the Safety of Air Navigation) tarafından birçok doküman yayımlanmıştır. Bunların incelenmesinin de Hava Seyrüsefer Sağlayıcıları için faydalı olacağı düşünülmektedir.

Ülkemizde de Esenboğa, Atatürk ve Antalya Havalimanlarına ASMGCS sistemleri kurulmakta ve bu sistemle birlikte MLAT da alt sistem olarak kurulmaktadır. Muhtemelen de önümüzdeki yıllarda ihtiyaca göre birçok havaalanına kurulacaktır.

Maliyet açısından bakıldığında; radarlara kıyasla beşte bir oranda daha az bir fiyata mal olduğundan dolayı WAM sistemlere yönelim artacaktır. Bunun sonucunda kullanımı yaygınlaştıkça bu konuda yapılan çalışmalar da artacaktır. Ülkemizde de WAM sistemler üzerine yapılan çalışmalar hızlanacaktır. Bu tez çalışmasının MLAT sistemler konusunda temel bir çalışma olduğu düşünülmektedir.

KAYNAKLAR

1. Neven W.H.L., Quilter T.J., Weeden R., Hogendoorn R.A., "Wide Area Multilateration Study Report on EATMI TRS 131/04", ***EUROCONTROL, Brussels,*** 12-21 (2005).
2. Soyertem H., "ATS Gözetim Sistemleri ve Hizmetleri V. Basım", **DHMİ Seyrüsefer Dairesi Başkanlığı,** Ankara, 7-9,39 (2009).
3. Oktal H., Yaman K., "Haberleşme, Seyrüsefer, İzleme ve Hava Trafik Yönetim Teknolojisi (CNS/ATM) ve Bu Sistemin Türk Havasahasına Uygulaması", ***Havacılık ve Uzay Teknolojileri Dergisi,*** 1 (3): 39-47 (2004).
4. Callen J., "Multilateration: Radar's Replacement?", ***Avionics,*** 30-34 (2007).
5. Bucher R., Mısra D., "A Synthesizable VHDL Model of the Exact Solution for Three-dimensional Hyperbolic Positioning System", ***VLSI Design,*** 15 (2): 507-513 (2002).
6. Brown A.S., Hardiman D.F., Carter D.R., "Nonlinear Estimation Techniques for Positioning Using Multilateration", ***ION GNSS 18th International Technical Meeting of the Satellite Division,*** Long Beach CA, 553-559 (2005).
7. Trofimova Y., "Multilateration Error Investigation and Classification. Error Estimation", ***Transport and Telecommunication,*** 8 (2): 28-37 (2007).
8. Arjun S., Jibin M., Mihun R.M.N., Navin M.S., Rajeesh R., "Tracking and Positioning System Main Project Design Report", ***Amrita Institude of Technology&Science, Amritapuri,*** 1-12 (2007).
9. Wells D.E., Krakiwsky E.J., "The Method of Least Squares", ***Geodesy and Geomatics Engineering University of New Brunswick, Frederiction,*** 101-116 (1997).
10. İnternet: Creativerge, "Multilateration & ADS-B Reference Guide" http://www.multilateration.com (2008).
11. Jane's Airport Review, "MLAT Systems Gain Ground", 19 (2): 20 (2007).
12. Air Traffic Management, "Multilateration: The Challenge Ahead", 2: 26-29 (2007).
13. ICAO Journal, "Multilateration Technology is Well Suited to a Wide Range of Applications", 62 (3): 12-14,32-33 (2007).

EKLER

EK-1. Ezplot.m çizdirme fonksiyonu

```
function hh = ezplot(varargin)
%EZPLOT   Easy to use function plotter
%   EZPLOT(FUN) plots the function FUN(X) over the default domain
%   -2*PI < X < 2*PI, where FUN(X) is an explicitly defined function of X.
%
%   EZPLOT(FUN2) plots the implicitly defined function FUN2(X,Y) = 0 over
%   the default domain -2*PI < X < 2*PI and -2*PI < Y < 2*PI.
%
%   EZPLOT(FUN,[A,B]) plots FUN(X) over A < X < B.
%   EZPLOT(FUN2,[A,B]) plots FUN2(X,Y) = 0 over A < X < B and A < Y < B.
%
%   EZPLOT(FUN2,[XMIN,XMAX,YMIN,YMAX]) plots FUN2(X,Y) = 0 over
%   XMIN < X < XMAX and YMIN < Y < YMAX.
%
%   EZPLOT(FUNX,FUNY) plots the parametrically defined planar curve FUNX(T)
%   and FUNY(T) over the default domain 0 < T < 2*PI.
%
%   EZPLOT(FUNX,FUNY,[TMIN,TMAX]) plots FUNX(T) and FUNY(T) over
%   TMIN < T < TMAX.
%
%   EZPLOT(FUN,[A,B],FIG),
EZPLOT(FUN2,[XMIN,XMAX,YMIN,YMAX],FIG), or
%   EZPLOT(FUNX,FUNY,[TMIN,TMAX],FIG) plots the function over the
%   specified domain in the figure window FIG.
%
%   EZPLOT(AX,...) plots into AX instead of GCA or FIG.
%
%   H = EZPLOT(...) returns handles to the plotted objects in H.
%
%   Examples:
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
%   The easiest way to express a function is via a string:
%     ezplot('x^2 - 2*x + 1')
%
%   One programming technique is to vectorize the string expression using
%   the array operators .* (TIMES), ./ (RDIVIDE), .\ (LDIVIDE), .^ (POWER).
%   This makes the algorithm more efficient since it can perform multiple
%   function evaluations at once.
%     ezplot('x.*y + x.^2 - y.^2 - 1')
%
%   You may also use a function handle to an existing function. Function
%   handles are more powerful and efficient than string expressions.
%     ezplot(@humps)
%     ezplot(@cos,@sin)
%
%   EZPLOT plots the variables in string expressions alphabetically.
%     subplot(1,2,1), ezplot('1./z - log(z) + log(-1+z) + t - 1')
%   To avoid this ambiguity, specify the order with an anonymous function:
%     subplot(1,2,2), ezplot(@(z,t)1./z - log(z) + log(-1+z) + t - 1)
%
%   If your function has additional parameters, for example k in myfun:
%     %----------------------%
%     function z = myfun(x,y,k)
%     z = x.^k - y.^k - 1;
%     %----------------------%
%   then you may use an anonymous function to specify that parameter:
%     ezplot(@(x,y)myfun(x,y,2))
%
%   See also EZCONTOUR, EZCONTOURF, EZMESH, EZMESHC, EZPLOT3,
EZPOLAR,
%            EZSURF, EZSURFC, PLOT, VECTORIZE, FUNCTION_HANDLE.
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
%   Copyright 1984-2008 The MathWorks, Inc.
%   $Revision: 1.43.4.18 $  $Date: 2008/12/04 22:40:56 $

% Parse possible Axes input
[cax,args,nargs] = axescheck(varargin{:});

f = args{1};
args = args(2:end);

if ~ischar(f) && ~isa(f,'inline') && ~isa(f,'function_handle')
    error(id('InvalidExpression'),...
          'Input must be a string expression, function name, function handle, or
INLINE object.');
end
twofuns = 0;
if (nargs > 1)
  twofuns = (ischar(args{1}) || isa(args{1},'inline') ...
                          || isa(args{1}, 'function_handle'));
  if (length(args)>1 && length(args{2})<=1)
     twofuns = 0;
  end
end

% Place f into "function" form (inline).
if (twofuns)
  [f,fx0,varx] = ezfcnchk(f,0,'t');
else
  [f,fx0,varx] = ezfcnchk(f);
end
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
vars = varx;
nvars = length(vars);
if isa(f,'function_handle') && nvars==0
    nvars = nargin(f);  % can determine #args without knowing their names
end
labels = {fx0};
if ~iscell(f), f = {f}; end

if (twofuns)
  % Determine whether the two input functions have the same
  % independent variable.  That is, in the case of ezplot(x,y),
  % check that x = x(t) and y = y(t).  If not (x = x(p) and
  % y = y(q)), reject the plot.
  [fy,fy0,vary] = ezfcnchk(args{1},0,'t');
  nvars = max(nvars, length(vary));
  if isa(fy,'function_handle') && isempty(vary)
    nvars = max(nvars,nargin(fy));
  end
  f{2} = fy;
  labels{2} = fy0;

  % This is the case of ezplot('2','f(q)') or ezplot('f(p)','3').
  if isempty(varx) || isempty(vary)
    vars = union(varx,vary);
  end
end

vars = vars(~cellfun('isempty',vars));
if isempty(vars)
  if (twofuns)
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
      vars = {'t'};
    else
      if (nvars == 2)
        vars = {'x' 'y'};
      else
        vars = {'x'};
      end
    end
end
nvars = max(nvars, length(vars));
ninputs = length(args);

if (ninputs==1 && ~twofuns)
  if length(args{1}) == 4 && nvars == 2
    V = args;
    args{1} = [V{1}(1),V{1}(2)];
    args{2} = [V{1}(3),V{1}(4)];
  end
  % ezplot(f,[xmin,ymin]) covered in the default setting.
end

if ~twofuns
 switch nvars
  case 1
   % Account for variables of [char] length > 1
   [hp,cax] = ezplot1(cax,f{1},vars,labels,args{:});
   title(cax,texlabel(labels),'interpreter','tex');
   if ninputs>0 && isa(args{1},'double') && length(args{1}) == 4
     axis(cax,args{1});
   elseif ninputs > 1 && isa(args{2},'double') && ...
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
        length(args{2}) == 4
      axis(cax,args{2});
    end
   case 2
    hp = ezimplicit(cax,f{1},vars,labels,args{:});
   otherwise
    if (isa(f,'function_handle'))
      fmsg = func2str(f);
    else
      fmsg = char(f);
    end
    error(id('NonXYPlot'),'%s cannot be plotted in the xy-plane.',fmsg);
  end
else
   hp = ezparam(cax,f{1},f{2},vars,labels,args{2:end});
end

if nargout > 0
    hh = hp;
end

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

function [hp,newcax] = ezimplicit(cax,f,vars,labels,varargin)
% EZIMPLICIT Plot of an implicit function in 2-D.
%    EZIMPLICIT(cax,f,vars) plots in cax the string expression f
%    that defines an implicit function f(x,y) = 0 for x0 < x < x1
%    and y0 < y < y1, whose default values are x0 = -2*pi = y0
%    and x1 = 2*pi = y1.  The arguments of f are listed in vars and
%    a non-vector version of the function expression is in labels.
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
%   EZIMPLICIT(cax,f,vars,labels,[x0,x1]) plots the implicit function
%   f(x,y) = 0 for x0 < x < x1, x0 < y < x1.
%
%   EZIMPLICIT(cax,f,vars,labels,[x0,x1],[y0,y1]) plots the implicit
%   function f(x,y) = 0 for x0 < x < x1, y0 < y < y1.
%   In the case that f is not a function of x and y
%   (rather, say u and v), then the domain endpoints [u0,u1]
%   [v0,v1] are given alphabetically.
%
%   [HP,NEWCAX] = EZIMPLICIT(...) returns the handles to the plotted
%   objects in HP, and the axes used to plot the function in NEWCAX.

% If f is created from a string equation f(x,y) = g(x,y), change
% the equal sign '=' to a minus sign '-'
eqnHasEqualSign = false;
if (isa(f,'inline') && ~isempty(findstr(char(f), '=')))
  symvars = argnames(f);
  f = char(f);
  f = [strrep(f,'=','-(') ')'];
  f = inline(f, symvars{:});
  eqnHasEqualSign = true;
end

% Choose the number of points in the plot
npts = 250;

fig = [];
switch length(vars)
case 0
  x = 'x'; y = 'y';
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
case 1
  x = vars{1}; y = 'y';
case 2
  x = vars{1}; y = vars{2};
otherwise
  % If there are more than 2 variables, send an error message
  W = {vars{1},vars{2}};
  error(id('NumericValues'),...
      'ezplot requires numeric values for %s',setdiff(vars,W));
end
% Define the computational space
switch (nargin-3)
case 1
  X = linspace(-2*pi,2*pi,npts);
  Y = X;
case 2
  if length(varargin{1}) == 1
    fig = varargin{1};
    X = linspace(-2*pi,2*pi,npts); Y = X;
  else
    X = linspace(varargin{1}(1),varargin{1}(2),npts);
    Y = X;
  end
case 3
  if length(varargin{1}) == 1
    fig = varargin{1};
    X = linspace(varargin{2}(1),varargin{2}(2),npts);
    Y = X;
  elseif length(varargin{2}) == 1 && length(varargin{1}) == 2
    fig = varargin{2};
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
    X = linspace(varargin{1}(1),varargin{1}(2),npts);
    Y = X;
  elseif length(varargin{2}) == 1 && length(varargin{1}) == 4
    fig = varargin{2};
    X = linspace(varargin{1}(1),varargin{1}(2),npts);
    Y = linspace(varargin{1}(3),varargin{1}(4),npts);
  else
    X = linspace(varargin{1}(1),varargin{1}(2),npts);
    Y = linspace(varargin{2}(1),varargin{2}(2),npts);
  end
end

[X,Y] = meshgrid(X,Y);
u = ezplotfeval(f,X,Y);

% Determine u scale so that "most" of the u values
% are in range, but singularities are off scale.

u = real(u);
uu = sort(u(isfinite(u)));
N = length(uu);
if N > 16
  del = uu(fix(15*N/16)) - uu(fix(N/16));
  umin = max(uu(1)-del/16,uu(fix(N/16))-del);
  umax = min(uu(N)+del/16,uu(fix(15*N/16))+del);
elseif N > 0
  umin = uu(1);
  umax = uu(N);
else
  umin = 0;
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
  umax = 0;
end
if umin == umax, umin = umin-1; umax = umax+1; end

% Eliminate vertical lines at discontinuities.

ud = (0.5)*(umax - umin); umean = (umax + umin)/2;
[nr,nc] = size(u);
% First, search along the rows . . .
for j = 1:nr
  k = 2:nc;
  kc = find( abs(u(j,k) - u(j,k-1)) > ud );
  ki = find( max( abs(u(j,k(kc)) - umean), abs(u(j,k(kc)-1) - umean) ) );
  if any(ki), u(j,k(kc(ki))) = NaN; end
end
% . . . then search along the columns.
for j = 1:nc
  k = 2:nr;
  kr = find( abs(u(k,j) - u(k-1,j)) > ud );
  kj = find( max( abs(u(k(kr),j) - umean), abs(u(k(kr)-1,j) - umean) ) );
  if any(kj), u(k(kr(kj)),j) = NaN; end
end

% First check if cax was specified (strongest specification for plot axes)
if isempty(cax)
   % Now allow the fig input to be honored
   cax = determineAxes(fig);
end

% EZPLOT calls the 'v6' version of CONTOUR, and temporarily modifies global
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
% state by turning the MATLAB:contour:DeprecatedV6Argument and
% MATLAB:contour:IgnoringV6Argument warnings off and on again.
oldWarn(1) = warning('off','MATLAB:contour:DeprecatedV6Argument');
oldWarn(2) = warning('off','MATLAB:contour:IgnoringV6Argument');
try
   [cmatrix,hp] = contour('v6',cax,X(1,:),Y(:,1),u,[0,0],'-'); %#ok
catch err
   warning(oldWarn); %#ok<WNTAG>
   rethrow(err);
end
warning(oldWarn); %#ok<WNTAG>

if (isa(x,'function_handle'))
  xmsg = func2str(x);
else
  xmsg = char(x);
end
if (isa(y,'function_handle'))
  ymsg = func2str(y);
else
  ymsg = char(y);
end
xlabel(cax,texlabel(xmsg)); ylabel(cax,texlabel(ymsg));
if eqnHasEqualSign
 title(cax,texlabel(labels{1}));
else
 title(cax,texlabel([labels{1},' = 0']));
end

newcax = cax;
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
%---------------------------------
function [hp,newcax] = ezparam(cax,x,y,vars,labels,varargin)
% EZPARAM Easy to use 2-d parametric curve plotter.
%   EZPARAM(cax,x,y,vars,labels) plots the planar curves r(t) = (x(t),y(t))
%   in cax.  The default domain in t [0,2*pi].  vars contains the common
%   argument of x and y, and labels contains non-vector versions of the
%   x and y expressions.
%
%   EZPARAM(cax,x,y,vars,labels,[tmin,tmax]) plots r(t) = (x(t),y(t)) for
%   tmin < t < tmax.
%
%   [HP,NEWCAX] = EZPARAM(...) returns the handles to the plotted
%   objects in HP, and the axes used to plot the function in NEWCAX.

fig = [];
N = length(vars);

Npts = 300;

% Determine the domains in t:
switch (nargin-3)
case 2
  T = linspace(0,2*pi,Npts);
case 3
  if length(varargin{1}) == 1
    fig = varargin{1};
    T = linspace(0,2*pi,Npts);
  else
    T = linspace(varargin{1}(1),varargin{1}(2),Npts);
  end
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
case 4
  if length(varargin{2}) == 1
    fig = varargin{2};
    T = linspace(varargin{1}(1),varargin{1}(2),Npts);
  elseif length(varargin{1}) == 1
    fig = varargin{1};
    T = linspace(varargin{2}(1),varargin{2}(2),Npts);
  else
    T = linspace(varargin{1},varargin{2},Npts);
  end
end

% First check if cax was specified (strongest specification for plot axes)
if isempty(cax)
   % Now allow the fig input to be honored
   cax = determineAxes(fig);
end

% Create plot
cax = newplot(cax);

switch N
  case 1 % planar curve
    X = ezplotfeval(x,T);
    Y = ezplotfeval(y,T);
    hp = plot(X,Y,'parent',cax);
    xlabel(cax,'x'); ylabel(cax,'y');
    axis(cax,'equal');
    title(cax,['x = ' texlabel(labels{1}), ', y = ' texlabel(labels{2})]);
  otherwise
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
    error('MATLAB:ezplot:ParametrizedSurface',...
        'Cannot plot parametrized surfaces.  Try ezsurf.')
end

newcax = cax;

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

function [hp,newcax] = ezplot1(cax,f,vars,labels,xrange,fig)
%EZPLOT1 Easy to use function plotter.
%   EZPLOT1(cax,f,vars,labels) plots a graph of f(x) into cax
%   where f is a string or a symbolic expression representing a
%   mathematical expression involving a single symbolic variable,
%   say 'x'.
%   vars is the name of the variable and labels is a non-vector
%   version of the function expression.
%   The range of the x-axis is approximately  [-2*pi, 2*pi]
%
%   EZPLOT1(cax,f,vars,labels,xmin,xmax) or EZPLOT(f,[xmin,xmax])
%   uses the specified x-range instead of the default [-2*pi, 2*pi].
%
%   EZPLOT1(cax,f,vars,labels,[xmin xmax],fig) uses the specified
%   figure number, fig, instead of cax.
%
%   [HP,NEWCAX] = EZPLOT1(...) returns the handles to the plotted
%   objects in HP, and the axes used to plot the function in NEWCAX.

% Set defaults
error(nargchk(4,6,nargin,'struct'));
if nargin < 5, xrange = [-2*pi 2*pi]; end
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
if ischar(xrange), xrange = eval(xrange); end
if nargin < 6, fig = ancestor(cax,'figure'); end
if nargin == 6
  if length(xrange) == 1
    xrange = [xrange fig];
  elseif ischar(fig)
    xrange = [xrange eval(fig)];
  elseif ~isempty(cax)
   fig = ancestor(cax,'figure');
  end
end

% Check for equations of the form "x=2"
if (isa(f,'inline') && ~isempty(findstr(char(f), '=')))
   error(id('NonExplicitFunction'),...
         ['The input string must be an expression.  ',...
         'Implicit functions of a single variable are not supported.']);
end

% First check if cax was specified (strongest specification for plot axes)
if isempty(cax)
   % Now allow the fig input to be honored
   cax = determineAxes(fig);
end

% Create plot
cax = newplot(cax);

warns = warning('off'); %#ok
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
% Sample on initial interval.
fig = ancestor(cax,'figure');
pixpos = hgconvertunits(fig,get(cax,'Position'),get(cax,'Units'),...
                        'pixels',get(cax,'Parent'));
% npts = # of pixels in the axis width.
npts = pixpos*[0;0;1;0];
t = (0:npts-1)/(npts-1);
xmin = min(xrange);
xmax = max(xrange);
x = xmin + t*(xmax-xmin);

% Get y values, and possibly also change f to be vectorized
[y,f,loopflag] = ezplotfeval(f,x);

k = find(abs(imag(y)) > 1.e-6*abs(real(y)));
if any(k), x(k) = []; y(k) = []; end
npts = length(y);
if isempty(y) && npts == 0
  warning(warns);
  warning('MATLAB:ezplot:NoRealValues', ...
        'Cannot plot %s:  This function has no real values.', ...
        labels{1});
  return
elseif loopflag
  % Warnings are off, so turn them on temporarily and issue a warning
  % message similar to what would have come from ezplotfeval.
  warning(warns);
  warning('MATLAB:ezplot:NotVectorized', ...
['Function failed to evaluate on array inputs; vectorizing the function may\n'...
'speed up its evaluation and avoid the need to loop over array elements.']);
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
  warning('off'); %#ok
end
% Reduce to an "interesting" x interval.

if (npts > 1) && (nargin < 5)
  dx = x(2)-x(1);
  dy = diff(y)/dx;
  dy(npts) = dy(npts-1);
  k = find(abs(dy) > .01);
  if isempty(k), k = 1:npts; end
  xmin = x(min(k));
  xmax = x(max(k));
  if xmin < floor(4*xmin)/4 + dx, xmin = floor(4*xmin)/4; end
  if xmax > ceil(4*xmax)/4 - dx, xmax = ceil(4*xmax)/4; end
  x = xmin + t*(xmax-xmin);
  y = ezplotfeval(f,x);
  k = find(abs(imag(y)) > 1.e-6*abs(real(y)));
  if any(k), y(k) = NaN; end
end

% Determine y scale so that "most" of the y values
% are in range, but singularities are off scale.

y = real(y);
u = sort(y(isfinite(y)));
npts = length(u);
if isempty(u)
  u = nan(size(x));
  npts = numel(x);
end
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
ymin = u(1);
ymax = u(npts);
if npts > 4
  del = u(fix(7*npts/8)) - u(fix(npts/8));
  ymin = max(u(1)-del/8,u(fix(npts/8))-del);
  ymax = min(u(npts)+del/8,u(fix(7*npts/8))+del);
end

% Eliminate vertical lines at discontinuities.

k = 2:length(y);
k = find( ((y(k) > ymax/2) & (y(k-1) < ymin/2)) | ...
        ((y(k) < ymin/2) & (y(k-1) > ymax/2)) );
if any(k), y(k) = NaN; end

% Plot the function

hp = plot(x,y,'parent',cax);
if ymax > ymin
  axis(cax,[xmin xmax ymin ymax])
else
  axis(cax,[xmin xmax get(cax,'ylim')])
end

xlabel(cax,texlabel(vars{1}));
title(cax,texlabel(labels{1}),'Interpreter','none')
warning(warns)

newcax = cax;
```

EK-1. (Devam) Ezplot.m çizdirme fonksiyonu

```
%-------------------------

function cax = determineAxes(fig)
% Helper function that takes the specified figure handle.  If the handle is
% not empty, find its current axes.  If it is empty, use the current axes.
if ~isempty(fig)
   % In case a figure handle was specified, but the figure does not exist,
   % create one.
   figure(fig);
   cax = gca(fig);
else
   % Neither cax nor fig was specified, so use gca
   cax = gca;
end

function str=id(str)
str = ['MATLAB:ezplot:' str];
```

EK-2. Ezimplot3.m çizdirme fonksiyonu

```
function h = ezimplot3(varargin)
% EZIMPLOT3    Easy to use 3D implicit plotter.
%   EZIMPLOT3(FUN) plots the function FUN(X,Y,Z) = 0 (vectorized or not)
%   over the default domain:
%   -2*PI < X < 2*PI, -2*PI < Y < 2*PI, -2*PI < Z < 2*PI.
%   FUN can be a string, an anonymous function handle, a .M-file handle, an
%   inline function or a symbolic function (see examples below)
%
%   EZIMPLOT3(FUN,DOMAIN)plots FUN over the specified DOMAIN instead of
the
%   default domain. DOMAIN can be vector
[XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX] or
%   vector [A,B] (to plot over A < X < B, A < Y < B, A < Z < B).
%
%   EZIMPLOT3(..,N) plots FUN using an N-by-N grid. The default value for
%   N is 60.

%   EZIMPLOT3(..,'color') plots FUN with color 'color'. The default value
%   for 'color' is 'red'. 'color' must be a valid Matlab color identifier.
%
%   EZIMPLOT3(axes_handle,..) plots into the axes with handle axes_handle
%   instead of into current axes (gca).
%
%   H = EZIMPLOT3(...) returns the handle to the patch object this function
%   creates.
%
% Example:
% Plot x^3+exp(y)-cosh(z)=4, between -5 and 5 for x,y and z
%
%   via a string:
```

EK-2. (Devam) Ezimplot3.m çizdirme fonksiyonu

```
% f = 'x^3+exp(y)-cosh(z)-4'
% ezimplot3(f,[-5 5])
%
%   via a anonymous function handle:
% f = @(x,y,z) x^3+exp(y)-cosh(z)-4
% ezimplot3(f,[-5 5])
%
%   via a function .m file:
%------------------------------%
% function out = myfun(x,y,z)
% out = x^3+exp(y)-cosh(z)-4;
%------------------------------%
% ezimplot3(@myfun,[-5 5])   or  ezimplot('myfun',[-5 5])
%
%   via a inline function:
% f = inline('x^3+exp(y)-cosh(z)-4')
% ezimplot3(f,[-5 5])
%
%   via a symbolic expression:
% syms x y z
% f = x^3+exp(y)-cosh(z)-4
% ezimplot3(f,[-5 5])
%
% Note: this function do not use the "ezgraph3" standard, like ezsurf,
% ezmesh, etc, does. Because of this, ezimplot3 only tries to imitate that
% interface. A future work must be to modify "ezgraph3" to include a
% routine for implicit surfaces based on this file
%
%   Inspired by works of:   Artur Jutan   UWO 02-02-98 ajutan@julian.uwo.ca
%   Made by:            Gustavo Morales   UC  04-12-09 gmorales@uc.edu.ve
```

EK-2. (Devam) Ezimplot3.m çizdirme fonksiyonu

```
%%% Checking & Parsing input arguments:
if ishandle(varargin{1})
   cax = varargin{1}; % User selected axes handle for graphics
   axes(cax);
   args{:} = varargin{2:end}; %ensuring args be a cell array
else
   args = varargin;
end
[fun domain n color] = argcheck(args{:});
%%% Generating the volumetric domain data:
xm = linspace(domain(1),domain(2),n);
ym = linspace(domain(3),domain(4),n);
zm = linspace(domain(5),domain(6),n);
[x,y,z] = meshgrid(xm,ym,zm);
%%% Formatting "fun"
[f_handle f_text] = fix_fun(fun); % f_handle is the anonymous f-handle for "fun"
                                  % f_text is "fun" ready to be a title
%%% Evaluating "f_handle" in domain:
try
  fvalues = f_handle(x,y,z);        % fvalues: volume data
catch ME
  error('Ezimplot3:Functions', 'FUN must have no more than 3 arguments');
end
%%% Making the 3D graph of the 0-level surface of the 4D function "fun":
h = patch(isosurface(x,y,z,fvalues,0)); % "patch" handles the structure...
                                        % sent by "isosurface"
isonormals(x,y,z,fvalues,h)% Recalculating the isosurface normals based...
                           % on the volume data
set(h,'FaceColor',color,'EdgeColor','none');
%%% Λditional graphic details:
```

EK-2. (Devam) Ezimplot3.m çizdirme fonksiyonu

```
xlabel('x');ylabel('y');zlabel('z');% naming the axis
alpha(0.7) % adjusting for some transparency
grid on; view([1,1,1]); axis equal; camlight; lighting gouraud
%%% Showing title:

%
%-----------------------------------------Sub-functions HERE---
function [f dom n color] = argcheck(varargin)
%ARGCHECK(arg) parses "args" to the variables "f"(function),"dom"(domain)
%,"n"(grid size) and "c"(color)and TRIES to check its validity
switch nargin
    case 0
        error('Ezimplot3:Arguments',...
        'At least "fun" argument must be given');
    case 1
        f = varargin{1};
        dom = [-2*pi, 2*pi]; % default domain: -2*pi < xi < 2*pi
        n = 60;              % default grid size
        color = 'red';       % default graph color
    case 2
        f = varargin{1};
        if isa(varargin{2},'double') && length(varargin{2})>1
          dom = varargin{2};
          n = 60;
          color = 'red';
        elseif isa(varargin{2},'double') && length(varargin{2})==1
          n = varargin{2};
          dom = [-2*pi, 2*pi];
          color = 'red';
        elseif isa(varargin{2},'char')
```

EK-2. (Devam) Ezimplot3.m çizdirme fonksiyonu

```
            dom = [-2*pi, 2*pi];
            n = 60;
            color = varargin{2};
        end
    case 3                % If more than 2 arguments are given, it's
        f = varargin{1};  % assumed they are in the correct order
        dom = varargin{2};
        n = varargin{3};
        color = 'red';    % default color
    case 4                % If more than 2 arguments are given, it's
        f = varargin{1};  % assumed they are in the correct order
        dom = varargin{2};
        n = varargin{3};
        color = varargin{4};
    otherwise
        warning('Ezimplot3:Arguments', ...
        'Attempt will be made only with the 4 first arguments');
        f = varargin{1};
        dom = varargin{2};
        n = varargin{3};
        color = varargin{4};
end
if  length(dom) == 2
   dom = repmat(dom,1,3);       %domain repeated in all variables
elseif length(dom) ~= 6
   error('Ezimplot3:Arguments',...
   'Input argument "domain" must be a row vector of size 2 or size 6');
end
%
%--------------------------------------
```

EK-2. (Devam) Ezimplot3.m çizdirme fonksiyonu

```
function [f_hand f_text] = fix_fun(fun)
% FIX_FUN(fun) Converts "fun" into an anonymous function of 3 variables (x,y,z)
% with handle "f_hand" and a string "f_text" to use it as title
types = {'char','sym','function_handle','inline'}; % cell array of 'types'
type = ''; %Identifing FUN object class
for i=1:size(types,2)
   if isa(fun,types{i})
      type = types{i};
      break;
   end
end
switch type
      case 'char' % Formatting FUN if it is char type. There's 2 possibilities:
      % A string with the name of the .m file
         if exist([fun,'.m'],'file')
            syms x y z;
            if nargin(str2func(fun)) == 3
               f_sym = eval([fun,'(x,y,z)']); % evaluating FUN at the sym point (x,y,z)
            else
               error('Ezimplot3:Arguments',...
               '%s must be a function of 3 arguments or unknown function',fun);
            end
            f_text = strrep(char(f_sym),' ',''); % converting to char and eliminating
spaces
            f_hand = eval(['@(x,y,z)',vectorize(f_text),';']); % converting string to
anonymous f_handle
         else
      % A string with the function's expression
            f_hand = eval(['@(x,y,z)',vectorize(fun),';']); % converting string to
anonymous f_handle
```

EK-2. (Devam) Ezimplot3.m çizdirme fonksiyonu

```
        f_text = strrep(fun,'.',''); f_text = strrep(f_text,' ',''); % removing
vectorization & spaces
      end
    case 'sym' % Formatting FUN if it is a symbolic object
      f_hand = eval(['@(x,y,z)',vectorize(fun),';']); % converting string to
anonymous f_handle
      f_text = strrep(char(fun),' ',''); % removing spaces
    case {'function_handle', 'inline'} % Formatting FUN if it is a function_handle or
an inline object
      syms x y z;
      if nargin(fun) == 3 && numel(symvar(char(fun))) == 3 % Determining if #
variables == 3
        f_sym = fun(x,y,z);   % evaluating FUN at the sym point (x,y,z)
      else
        error('Ezimplot3:Arguments',...
        '%s must be function of 3 arguments or unknown function',char(fun));
      end
      f_text = strrep(char(f_sym),' ','');  % converting into string to removing spaces
      f_hand = eval(['@(x,y,z)',vectorize(f_text),';']); % converting string to
anonymous f_handle
  otherwise
      error('First argument "fun" must be of type character, simbolic, function
handle or inline');
end
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: PAŞAOĞLU,Cengiz
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 09.10.1979 Trabzon
Medeni hali	: Evli
Telefon	: 0 (312) 204 26 85
Faks	: 0 (312) 222 60 05
e-mail	: cengiz.pasaoglu@dhmi.gov.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gazi Üniversitesi/ Elk.-Eln. Müh.	2002
Lise	Gölcük Anadolu Lisesi	1997

İş Deneyimi

Yıl	Yer	Görev
2002-2007	DHMİ Esenboğa Hv. Lim.	Hv. Trf. Kont.
2007-	DHMİ Gn. Md. lüğü	Ar-Ge Kont./Eln. Müh.

Yabancı Dil

İngilizce

Hobiler

Bilgisayar Teknolojileri, Yüzme, Futbol, Trabzonspor.