KARADENİZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GENİŞ BÖLGE İZLEME SİSTEMLERİNDE OLAY YAKALAMA ALGORİTMASI

YÜKSEK LİSANS TEZİ

Elektrik Elektronik Müh. Mehmet Emre TERZİ

Kasım 2019

TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GENİŞ BÖLGE İZLEME SİSTEMLERİNDE OLAY YAKALAMA ALGORİTMASI

Mehmet Emre TERZİ

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"ELEKTRİK YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 05 / 09 /2019
Tezin Savunma Tarihi : 01 / 11 /2019

Tez Danışmanı : Dr. Öğr. Üyesi Fatih Mehmet NUROĞLU

Trabzon 2019

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik-Elektronik Mühendisliği Anabilim Dalında
Mehmet Emre TERZİ Tarafından Hazırlanan

GENİŞ BÖLGE İZLEME SİSTEMLERİNDE OLAY YAKALAMA ALGORİTMASI

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 15 / 10 / 2019 gün ve 1823 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Engin Özdemir .....

Üye : Dr. Öğr. Üyesi Fatih Mehmet Nuroğlu .....

Üye : Dr. Öğr. Üyesi Mehmet Öztürk .....

Prof. Dr. Asim KADIOĞLU
Enstitü Müdürü

ÖNSÖZ

En vazgeçilmez enerji türümüz olan elektrik enerjisinin kesintisiz, güvenilir ve sürdürülebilir bir şekilde üretilmesi, iletilmesi ve dağıtılması, olan ve olacak olayların tespiti ve alınabilecek önlemlerin zamanında alınmasıyla mümkündür. Ülkemizin karanlığa gömülmemesi ve üretim tesislerimizin kesintiye uğramaması adına bu olayların tespiti için farklı yöntemler uygulanmaktadır. Bu amaçla tez çalışmamda frekans dinamikleri ile olay tespiti yapan algoritma geliştirilmiştir. Şebeke frekansını dinleyen, kaydeden ve bu bilgiler doğrultusunda olayları tespit eden Visual Studio ortamında Windows tabanlı c# dilinde yazılım geliştirilmiştir.

Bu tez çalışmasında danışmanlığımı yaparak bana bilgi ve düşünceleriyle yol gösteren, her daim beni destekleyen, ilgi ve tecrübelerini esirgemeyen değerli hocam Dr. Öğr. Üyesi Fatih M. NUROĞLU'na teşekkürlerimi sunarım. Her zaman sorularıma sabırla cevap veren Arş.Gör. Hatice OKUMUŞ'a ve bu süreç boyunca bana yardımcı olan tüm hocalarıma, dostlarıma; hayatım boyunca varlıkları ve destekleriyle bana güç veren ve her zaman yanımda olan aileme şükranlarımı sunarım.

Mehmet Emre TERZİ

Trabzon 2019

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Geniş Bölge İzleme Sistemlerinde Olay Yakalama Algoritması Geliştirilmesi” başlıklı bu çalışmayı baştan sona kadar danışmanım Dr. Öğr. Üyesi Fatih Mehmet NUROĞLU’nun sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 01/11/2019

Mehmet Emre TERZİ

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

GENİŞ BÖLGE İZLEME SİSTEMLERİNDE OLAY YAKALAMA ALGORİTMASI GELİŞTİRİLMESİ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik- Elektronik Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Fatih Mehmet NUROĞLU
2019, 75 Sayfa

Son zamanlarda ülkemizde ve dünyada yaşanan büyük ölçekli Sistem Çökmeleri güç sistemlerinin güvenlik ve istikrardan uzak olduğunu göstermiştir. Gerek olay esnasında gerekse olay sonrasında sistemin çökmesine sebep olan durumların analizinde bilgi eksikliği en önemli etken olarak ortaya çıkmıştır. Elektrik Güç Sistemlerinin en önemli parametrelerinden birisi frekans parametreleridir. Güç sistemi dinamiklerini doğru ve etkin değerlendirebilmek için frekans dinamiklerine ihtiyaç duyulmaktadır.

Enterkonnekte şebekelerde çok sayıda üretim ve tüketim noktaları birbirine senkron olduğundan dolayı arz ve talepte meydana gelen tüm değişimler frekansta dalgalanmalara sebep olmaktadır. Frekans verisindeki bu dalgalanmaya sebep olan normal koşulların meydana gelen olaylardan ayırt edilmesi için frekans verisinin filtrelenmesi ve bir algoritmadan geçmesi gerekmektedir. Bu çalışma ile frekans dinamikleri etkin bir şekilde takip edilmeye, uygun formatta veri tabanına kaydedilmeye ve geliştirilen algoritma sayesinde şebekede meydana gelen bozulmalar yakalanmaya çalışılmıştır.

Anahtar Kelimeler: Geniş Bölge İzleme, Frekans Dinamikleri, FNET

Master Thesis

SUMMARY

GENİŞ BÖLGE İZLEME SİSTEMLERİNDE OLAY YAKALAMA ALGORİTMASI GELİŞTİRİLMESİ

Karadeniz Tecnicial University
The Graduate School of Natural and Applied Sciences
Electrical-Electronics Engineering Graduate Program
Supervisor: Asst. Prof. Dr. Fatih Mehmet NUROĞLU
2019, 75 Pages

Major system breakdowns experienced in our national grid and worldwide indicate that power systems are lacking of reliability, robustness and security. The actions has taken to avoid the system breakdown and the inspections afterwards reveal the fact that the biggest cause of chain of disasters is again lack of technical knowledge. Frequency is one of the most important parameters of an electrical power grid. In order to evaluate power system dynamics correctly and efficiently, frequency parameters are paramountly needed.

Since many electrical generation and consumption points are synchronized with each other in interconnected networks, all changes in supply and demand cause fluctuations in the frequency. In order to separate the occurring events which cause these oscillations in the frequency data from the normal conditions, the frequency data must be filtered and passed through an algorithm. In this study, the frequency dynamics are monitored effectively, recorded in the database in the appropriate format and with the developed algorithm the disturbances in the network are detected.

Key Words: Wide Area Monitoring, Frequency, FNET

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

FDR	: Frequency Disturbance Recorder
Kgbi	: Geniş Bölge İzleme Katsayısı
MW	: Megawatt
TEİAŞ	: Türkiye Elektrik İletim Anonim Şirketi
GPS	: Global Positioning System
RAID	: Redundant Array of Inexpensive Disks
TCP	: Transmission Control Protocol
SQL	: Structured Query Language
IP	: Internet Protocol Address
UTC	: Eşgüdümlü Evrensel Zaman
IEEE	: Uluslararası Elektrik Elektronik Mühendisleri Enstitüsü
TUBİTAK	: Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
PMU	: Fazör Ölçü Birimi

1. GENEL BİLGİLER

1.1. Giriş

Teknolojinin, sanayinin gelişmesiyle beraber enerjiye olan talep artmaktadır. Son dönemlerde bu artış hızlanarak devam etmektedir. Bu durum enerji üretim, iletim ve dağıtım sistemlerinin önemini arttırmaktadır. Enerji üretildikten sonra tüketicilere güvenilir bir şekilde ulaştırılması için çeşitli dağıtım şekilleri kullanılmaktadır. Dağıtım şekillerinden en çok tercih edileni enterkonnekte şebekedir. Enterkonnekte sistem, bir ülkenin tamamının ya da belirli bölgelerinin elektrik enerji gereksinimlerini karşılayabilecek bir biçimde üretim ile tüketim merkezleri arasındaki enerji alışverişini sağlayan enerji taşıma sistemidir. Bu tip şebekelerde, o bölgedeki bütün elektrik üretim ve tüketim araçları büyük küçük ayrımı yapılmaksızın sisteme dahil edilmektedir. Bütün sistem üretimden tüketime kadar birbiriyle bağlantılı olduğu için, sistemde meydana gelen bütün olaylar sistemin genelinde birtakım etkilere sebep olmaktadır. Frekans dinamikleri yorumlanarak, sistemde meydana gelen bu değişimler ile olayın analizi etkin bir şekilde yapılabilmektedir. Elektrik Güç Sistemleri oldukça karmaşık bir yapıya sahiptir. Anlık olarak üretilen ve tüketilen enerji değişmektedir. Arz talep sürekli değiştiğinden enterkonnekte şebekede frekans dalgalanmaları meydana gelmektedir. Arz talep dengesinin sağlanamaması durumunda meydana gelen frekans dalgalanması, üretim santrallerinde ünitelerin devreden çıkmasına, belli bir bölgenin enerjisiz kalmasına ya da sistemin tamamen çökmesine sebep olabilir.

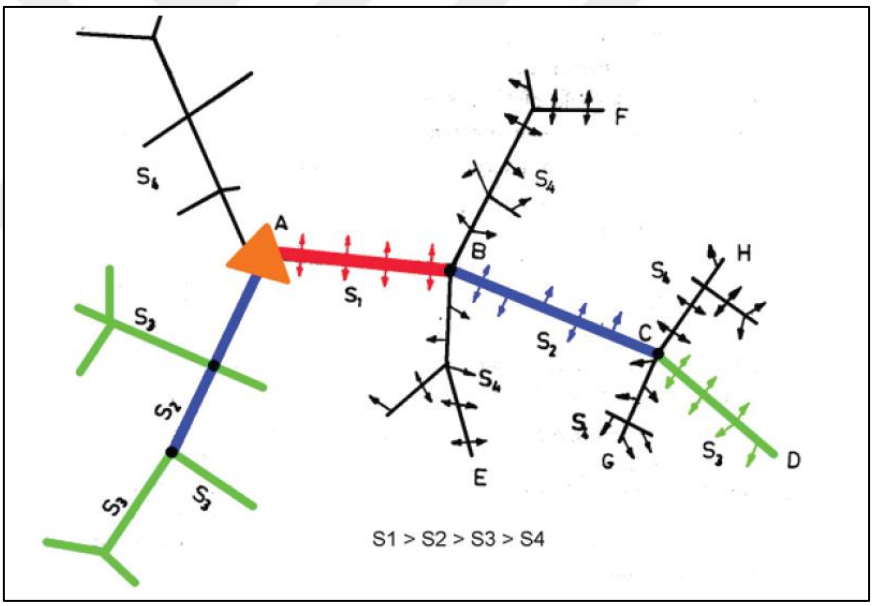
1.2. Şebeke Türleri

Elektriğin tüketildiği yerler genellikle üretim santrallerine uzaktır. Hatta bazı yerleşim yerinde elektrik üretim tesisi bulunmayabilir. Bu sebeple üretilen elektrik enerjisinin üretim tesisinden tüketim noktalarına taşınması gerekmektedir. Elektrik enerjisinin tüketicilere

ulaştırılması için üretim tesisleriyle tüketim birimlerinin birbirlerine iletim hatlarıyla bağlanmalıdır. Hatlar birleşerek kolları, kollar birleşerek şebekeyi oluşturur. Bu bağlantıların yapılma şekline göre farklı şebeke sistemleri geliştirilmiştir.

1.2.1. Dallı (Dalbudak) Şebekeler

Genellikle tek kaynaktan beslenen ve dağılım şekli ağaç dallarına benzeyen şebeke türüne denir. Dallı şebekelerde trafo, besleyeceği alanın yük bakımından ağırlık merkezine yerleştirilir. Trafodan tüketim noktalarına iletim hatları çekilir. Bu hatlar önce kalın kollara daha sonra da ince kollara ayrılarak son alıcıya kadar ulaşır.

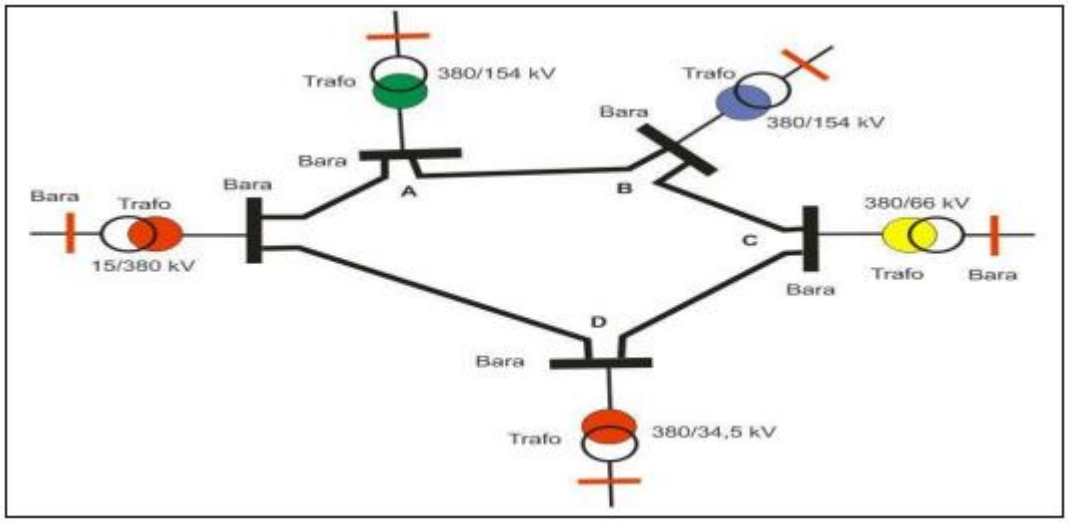


Şekil 1.1. Dallı Şebeke

Dallı şebekeler, maliyetinin ucuz olması, bakım ve işletmelerinin kolay olması ve oluşan arızaların bakımlarının kolay olması sebebiyle tercih edilebilir. Ancak dezavantajları da vardır. Dallı şebekelerde emniyet azdır. Bu şebekelerde koruyucu sistemler olarak sigortalar, aşırı akım röleli şalterler kullanılır. Bir arıza olduğunda çok sayıda abone enerjisiz kalabilir. Dallı şebekelerde hatlar aynı gerilim seviyesine sahip değildir. Şebekede bulunan trafoya olan mesafeye göre gerilim seviyesi değişiklik göstermektedir.

1.2.2. Ring (Halka) Şebeke

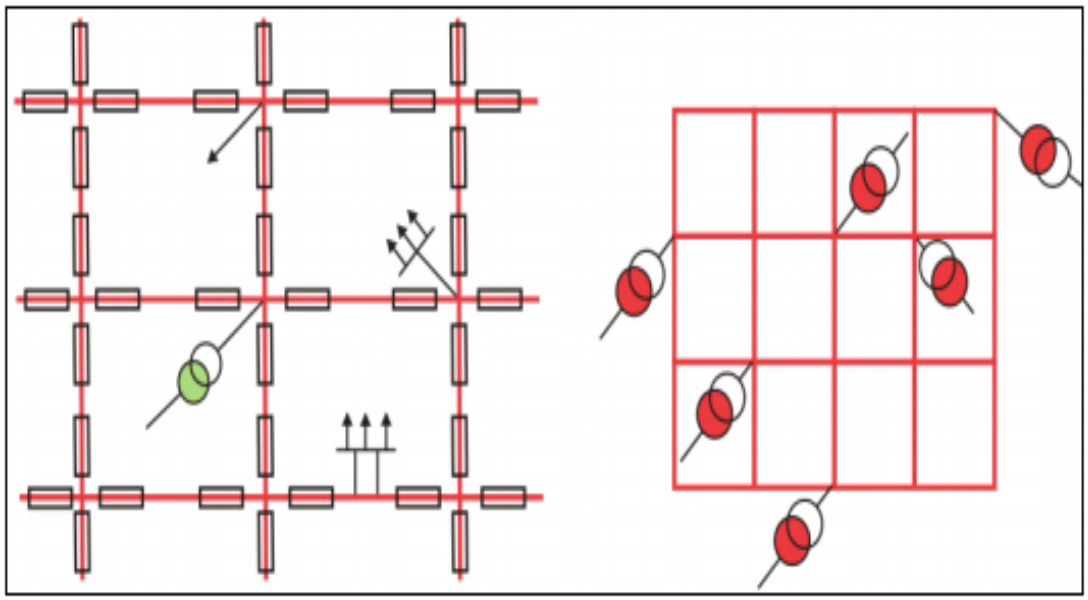
Ring şebekede birden fazla trafo bulunmaktadır ve bu trafolar birbirlerine paralel şekilde bağlanmıştır. Büyük ölçekli alanlarda uygulanmaktadır. Birden fazla trafo birbirine paralel bağlandığı için herhangi bir arıza durumunda dahi kısmi kesinti olur. Arızalı bölgenin devre dışı bırakılması sigorta ya da özel koruma elemanlarıyla sağlanır. Arızalı bölgelerin birbirini etkilemesi bu şekilde önlenmiş olur. Tabi ki bu denli bir tesisin maliyeti diğer türlere göre daha pahalıdır.



Şekil 1.2. Ring Şebeke

1.2.3. Ağ (Gözlü) Şebeke

Şehir, kasaba, köy ve sanayi merkezlerinde uygulanabilen beslemenin birden fazla trafo ile yapıldığı ve alıcıları besleyen hatların bir ağ gibi örülerek gözlerin oluşturulduğu şebeke tipine ağ şebeke denir. Ağ şebekeler de ring şebekeler gibi beslemenin sürekli yapılabildiği, arızanın sadece arıza olan yeri etkilediği bir sistemdir. Arıza olduğunda arızalı kısım sigortalar veya özel koruma elemanları ile devre dışı bırakılır. Diğer kısımların enerjisi kesilmez. Bazı ağ şebekelerde besleme bir yerden yapılır. Bu durumda yine kesintisiz enerji verebilir. Fakat trafo arıza yaptığında şebekenin tamamı enerjisiz kalır. [1]



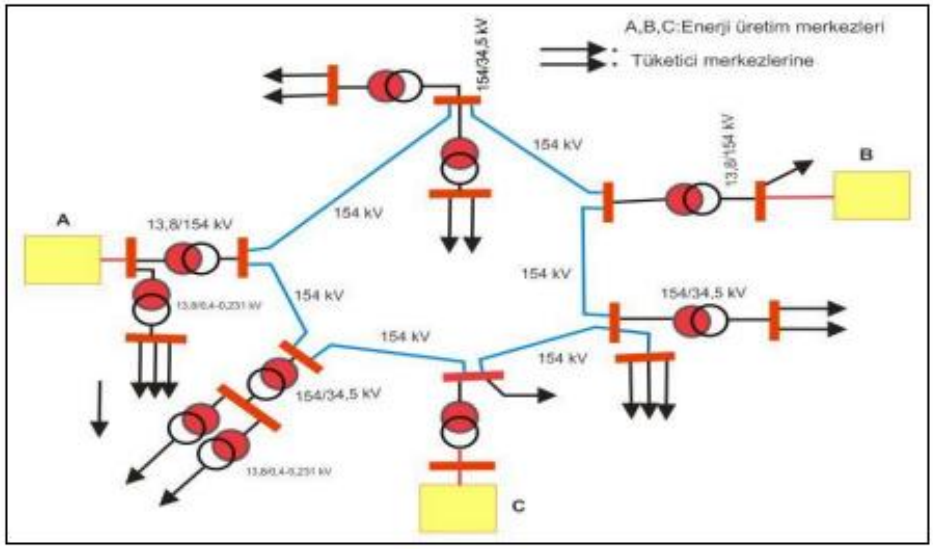
Şekil 1.3. Ağ (Gözlü) Şebeke

1.2.4. Enterkonnekte Şebeke

Günümüz teknolojisinde büyük güçlerde elektriğin depolanamaması sebebiyle elektriğin anlık olarak güvenli bir şekilde üretilip aynı zamanda tüketilmesi gerekmektedir. Arz talepteki bu zorunlu dengenin bozulmaması, sistemde meydana gelen arızaların etkisinin en aza indirgenmesi ve üretim tesislerinde meydana gelen arıza sonucu santrallerin beslediği bölgelerin enerjisiz kalmaması için bütün santrallerin birbirine paralel bağlandığı şebeke türüne enterkonnekte şebeke denmektedir.

Enterkonnekte şebekenin bu avantajlarının yanı sıra dezavantajları da bulunmaktadır. Özellikle kısa devre akımının çok yüksek olması en önemli dezavantajıdır. Bu sebeple kısa devre akımlarının hesaplanması büyük önem arz etmektedir.

Enterkonnekte şebekede, tüm üretim ve tüketim birimleri birbirleriyle bağlantılı olduğundan dolayı, ortak bir enerji piyasasının oluşmasına da zemin oluşturmaktadır. Bu sayede dengeli bir enerji piyasası oluşmakta ve dolaylı olarak sistemin güvenliğine olumlu katkı sağlamaktadır.



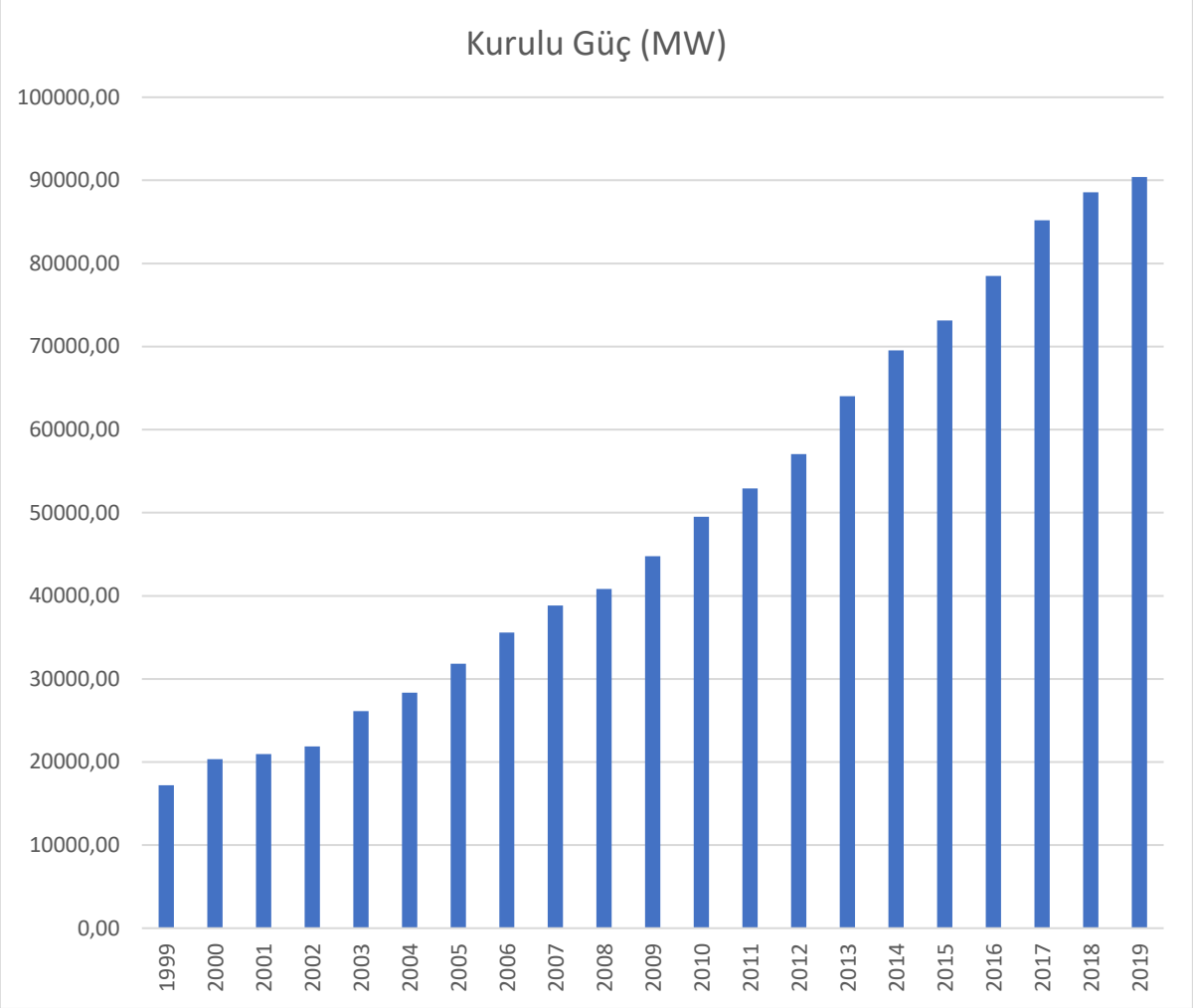
Şekil 1.4. Enterkonnekte Şebeke Yapısı

1.3. Türkiye Elektrik Şebekesi

Günümüzde enerji bir ülkenin ekonomisinin etkileyen temel taşlardan birisi haline gelmiştir. Günümüzde enerji üretim ve ticareti küresel boyutta gerçekleşmektedir. Globalleşen dünyamızda enterkonnekte şebekelerin de sayesinde uluslararası ilişkilerde de önemli bir güç haline gelmiştir. Bu sebeple bir ülkenin gelişimi aynı zamanda enerjide gelişim süreçlerine de bağlıdır. Gerek yenilenebilir enerji türleri gerekse yeraltı ve üstü enerji kaynaklarıyla enerji üretimini sürekli olarak arttırmak, ekonomik olarak dışa bağımlılığı önemli seviyede azaltmaktadır.

Ülkemizde elektrik üretimi 1902 yılında Tarsus'ta kurulan 2 kW gücündeki dinamo ile başlamıştır. İlk kayda değer üretim tesisi 1914 yılında kurulan Silahtarağa Termik Santralidir. İstanbul şehrinin 3 noktasına elektrik vermeye başlayan Silahtarağa Elektrik Santralı, Osmanlı devletinin ilk kent ölçekli ve termik santralı oldu.

Türkiye'de elektrik üretimi 1930'lu yıllara kadar yabancı işletmeciler tarafından genellikle küçük çaplı olarak kurulmuş ve bölgesel elektrik enerjisi üretimi ve dağıtımı şeklinde olmuştur. 1935 yılında ülkemiz 126,2 MW kurulu gücüne sahipti.



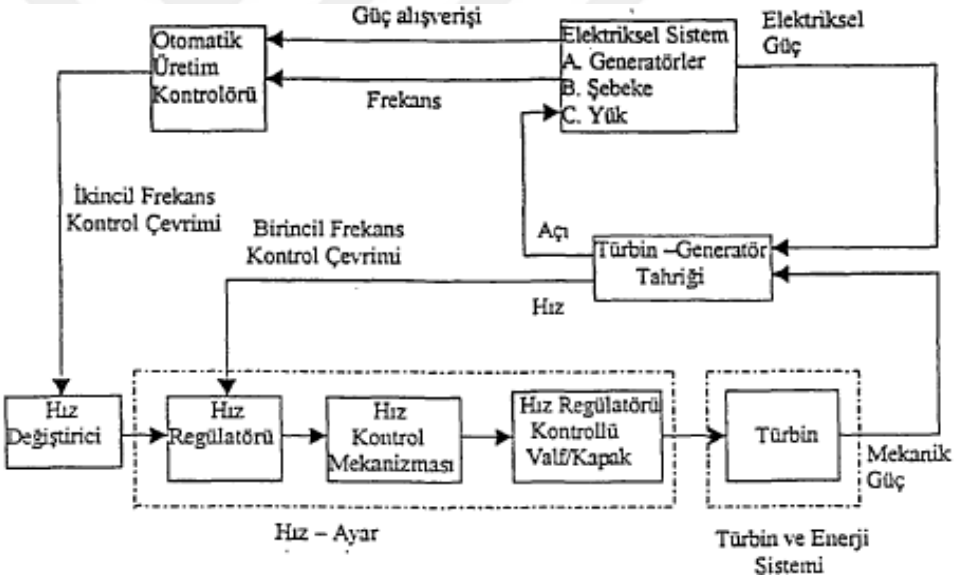
Şekil 1.5. Türkiye Kurulu Gücünün Yıllara Göre Gelişimi

1.4. Sistemde Meydana Gelen Olaylar

Mekanik enerji, elektrik enerjisi ve kayıpların toplamını ifade eder. Buhar, su, rüzgar gibi enerjiler ile türbinde üretilen mekaniksel moment (T_m) dönüş hızını artıran etki göstermektedir. Buna karşılık yükün oluşturduğu T_e elektriksel moment hızı azaltacak yönde etki etmektedir. Bu iki moment eşit olduğunda sistem sabit hızda w=w0 olmaktadır. Elektriksel yük artırıldığında $T_e > T_m$ olmakta ve generatör yavaşlamaktadır. Bu durumda generatör yeniden hızlandırılır, aksi durumda generatör yavaşlatılır. Enterkonnekte şebekede, yük hiçbir zaman sabit olmadığından bütün bu işlemler güç sistemlerinde sürekli olarak tekrarlanır.

$$P_m = P_e + P_{kayıp} \quad (1)$$

Enterkonnekte şebekede meydana gelen olaylar sonrası T_e ve T_m farklılaşacaktır. Bu durum generatör hızının artmasına veya azalmasına ve frekansın değişimine sebep olacaktır. Hız regülatörleri tepki vererek yük frekans kontrolünü sağlayacaktır. Yük frekans kontrolünün hedefi çok bölgeli enterkonnekte sistemlerdeki kararlı durum hatalarını sıfıra çekmek ve sıfırda sürdürmektir. [2]



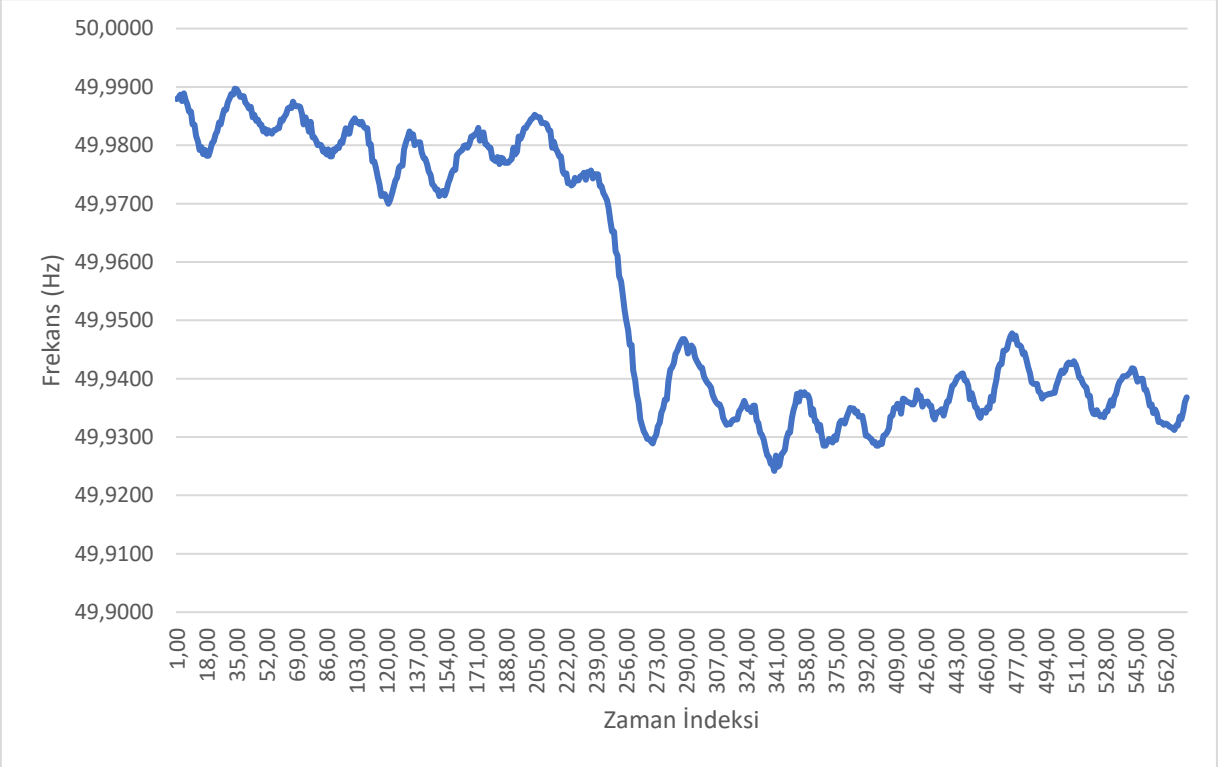
Şekil 1.6. Çok kontrol bölgeli bir sistemde otomatik yük-frekans kontrolünün birincil ve ikincil çevrimleri [3]

Primer ve sekonder frekans tutan santraller olaya tepki gösterecektir. Bu tepki yük alma ve yük atma şeklinde olacaktır. Elektrik enerjisindeki bu değişim generatörlerin dönme hızını yani frekansı etkileyecektir. Elektrik üretim tesislerinde bulunan hız regülatörleri bu değişime tepki göstererek mekanik enerjide ayarlama yapacaktır. Tüm bu olaylar frekans üzerinde etkiler meydana getirecektir. Tez çalışmasında bu değişim üzerinden olay yakalama algoritması geliştirilmiştir.

1.4.1. Generatör Devre Dışı Kalma Olayı (Generatör Trip)

İletim hattında ya da generatörün kendisinde meydana gelen arıza ve hatalardan kaynaklı olumsuz etkilerin en aza indirgenmesi amacıyla kontrol ve koruma sistemleri bulunmaktadır. Bu sistemlerin temelini röleler ve sensörler oluşturmaktadır. Arızanın tespiti sonrası en kısa sürede enerji kesme işlemini gerçekleştirerek generatörün zarar görmesini engellerler.

Üretimde olan generatörün koruma elemanları tarafından aniden devreden çıkarılmasıyla, enterkonnekte şebekede dalgalanmalar meydana gelir. Bu dalgalanmanın büyüklüğü, devreden çıkan güç ile sistemde o an bulunan gücün arasındaki orana göre doğrusallık göstermektedir.



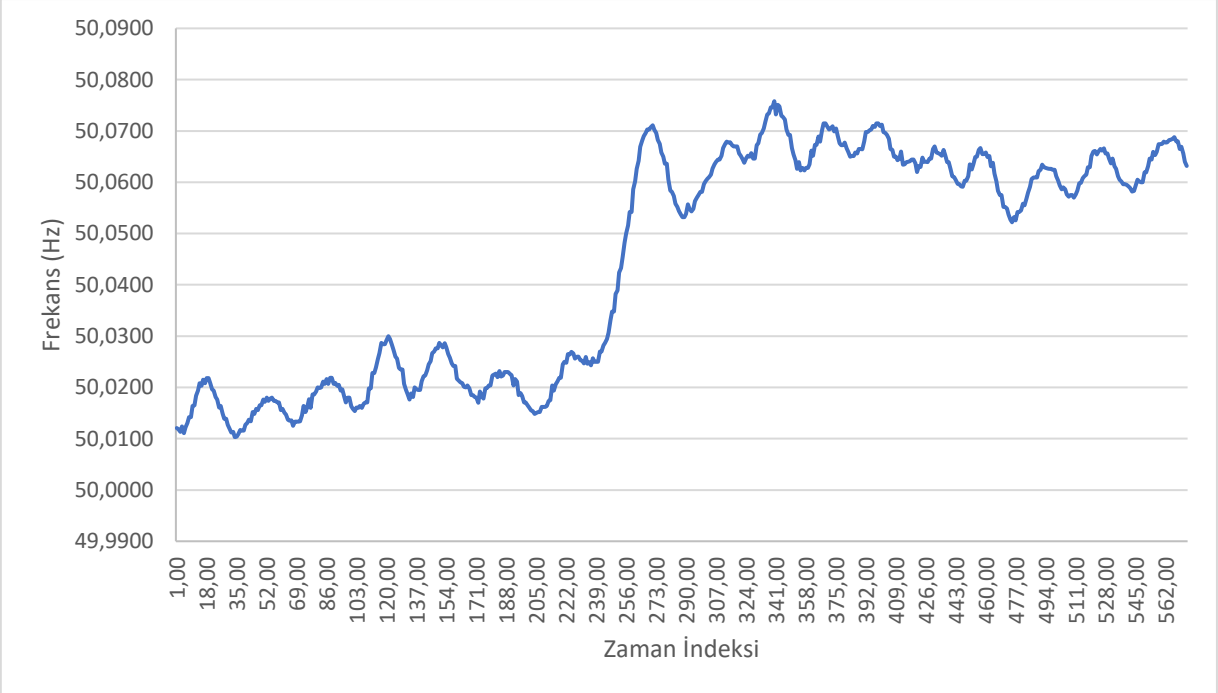
Şekil 1.7. Generatör Devre Dışı Kalma Olayı Esnasında Frekans Değişimi

Grafikte de görüleceği üzere arz tarafında yaşanan problem sonrası frekans önce yükselmekte ve sonrasında ani düşüş ve toparlanma sürecine girmektedir.

1.4.2. Yük Devre Dışı Kalma Olayı (Load Trip)

Yük devre dışı kalma olayı, enterkonnekte şebekeden aniden yük devre dışı kalması olayıdır. Bazen şebekede meydana gelen arıza sonrası yük devre dışı kalırken bazen de ülke çapında sistemin çökmesini önlemek için son çare olarak kullanılan bir yöntemdir. Enterkonnekte şebekedeki talebi karşılayabilecek üretim kapasitesi olmadığında, elektrik sistemi dengesiz hale gelir ve bu durum ülke genelinde 31 Mart 2015'te yaşadığımız gibi sistem çökmesine neden olabilir. Sistemin tekrardan ayağa kaldırılması zaman alacağından bu süre zarfında ciddi ekonomik kayıplar meydana gelecektir.

Arz talep dengesini sağlayabilmek adına TEİAŞ tarafından yük tahmini yapılır ve bu tahmin doğrultusunda üretim santrallerinden enerji tedariki yapılır. Yük tahminindeki sapmaların ve üretim tesislerinde meydana gelen dengesizlik üretimlerinin dengelenmesi için TEİAŞ Yük Tevzi birimleri tarafından üretim tesislerine Yük Al ve Yük At talimatları gönderilmektedir. Dengesizliğin önlenemediği ve üretimin yetersiz kaldığı durumlarda yük atması olayı meydana gelir.



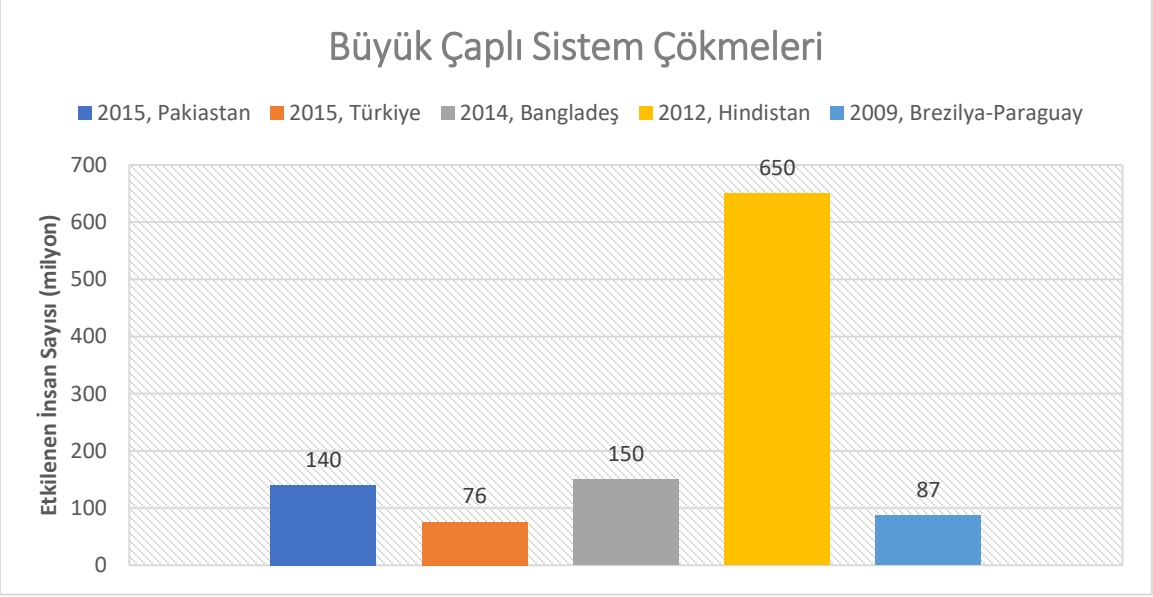
Şekil 1.8. Yük Devre Dışı Kalma Olayı Esnasında Frekans Değişimi

Grafikte de görüleceği üzere talep tarafında yaşanan problem sonrası frekans önce yükselmekte ve sonrasında ani düşüş ve toparlanma sürecine girmektedir.

1.4.3. Sistem Çökmesi (Blackout)

Elektrik güç sistemleri; kesintisiz, kaliteli, güvenilir ve ekonomik elektrik üretimi üzerine kurulmaktadır. Arızalar, bakım planlama hataları, işletme hataları, doğal afetler gibi durumlar güç sistemlerinde olumsuzluklara sebep olmaktadır. Bu olumsuzlukların büyük çapta gerçekleşmesi durumunda enterkonnekte şebekede arz talep dengesi engellenemeyecek boyutlara ulaşabilir ve şebekenin tamamında elektrik kesintisine yani sistem çökmesine sebep olabilir.

31 Mart 2015 tarihinde ülkemizde örneğini yaşamış olduğumuz sistem çökmeleri ciddi ekonomik kayıplara sebep olmaktadır. Son zamanlarda yaşanan sistem çökmelerini incelediğimizde milyonlarca insanın bu kesintilerden etkilendiğini ve milyarlarca dolar maddi kayıp meydana geldiğini görmekteyiz. [4]



Şekil 1.9. Büyük Çaplı Sistem Çökmeleri

1.5. Literatür Araştırması

Günümüzde elektriğin üretiminde, iletiminde ve dağıtımında kullandığımız alternatif akımı günümüzdeki şekliyle ilk olarak üreten Nocala Tesla'dır. Nicola Tesla, alternatif akımı 1886 yılında laboratuvarda üretmiştir. Alternatif akımın keşfinden sonra o yıllarda farklı bilim adamları da bu alanda çalışmalar yapmıştır.

Tennesse Üniversitesi Elektrik Elektronik ve Bilgisayar Departmanında kurulan Power IT Lab ekibi üyeleri Frekans İzleme Ağları (FNET) üzerine çalışmalar yapmaktadır. Tez çalışmamda kullandığım FDR cihazı bu ekip tarafından üretilip geliştirilmektedir. [5]

Dr. Yilu Liu ve ekibi şebeke gerilimi açı ölçümlerini kullanarak şebekede meydana gelen salınımları analiz etmiştir. Yapmış oldukları bu çalışma IEEE' de yayınlanmıştır.

Yingchen Zhang ve ekibi Geniş Alan Frekans İzleme Ağı (FNET) Mimarisi ve Uygulamaları alanında çalışmalar yapmıştır. Yapmış oldukları bu çalışma IEEE' de yayınlanmıştır. [6]

Chunchun XU ve ekibi, tez çalışmamda kullandığım FDR cihazından gelen bilgiler ile şebeke frekansındaki sapma ve frekans değişim bilgisi ile güç şebekesinin izleme ve koruma uygulamaları üzerine çalışmalar yapmıştır. [7]

TEİAŞ ve TUBİTAK tarafından enterkonnekte şebekeyi izleme adına geliştirilmiş Güç Kalitesi İzleme Cihazları bulunmaktadır. Geliştirilen bu cihazlardan bilgiler, TEİAŞ merkezleri tarafından izlenebilmektedir.

Güç sistemlerinin frekansı, bir güç şebekesinin durumunun önemli bir göstergesidir. Normal şartlarda enterkonnekte şebekeye bağlı tüm sistemde frekans aynı olmalıdır. Dolayısıyla yük akışı içerisinde arz talep arasındaki dengesizliklerden dolayı frekansta meydana gelen salınım tüm şebeke sisteminde aynı şekilde ilerlemektedir. Yük atması sonrası UPS ve yedek jeneratör üzerinden beslenen sistemlerde enterkonnekte şebekeye bağlantı olmadığı için farklı bir frekans salınımı meydana gelecektir. Jiahui Guo ve ekibi bu dalgalanma farklılığını inceleyerek bölgesel karartma ve yük atması gibi olayların yakalanması üzerine bir algoritma geliştirmiştir. [8]

S. Sterpu, yaptığı çalışmasında arz talep dengesizliğinden kaynaklanan frekans dalgalanmasının 50 mHz'den daha büyük olduğu noktanın çevresinde şebekede meydana gelen bir olay ihtimaline yönelik çalışma yapmış ve ihtimal bölgesi olarak 50 mHz lik değişimin olduğu frekans değerinin 200 değer öncesini ve 300 değer sonrasını almıştır. Belirlemiş olduğu bu çerçeve üzerinde algoritma geliştirerek olay yakalama yoluna gitmiştir. [9]

Do-In Kim, enterkonnekte şebekedeki geçici güç salınımı olaylarını incelemek için azaltılmış iki alanlı bir güç sistemi modeli kullanmıştır. Olay tespitinde gerilim ve frekans verilerini birlikte kullanmıştır. Gerilim ve frekansı izleyerek olay sınıflandırmasının teorik arka planı, gerçek güçteki değişimin frekansı etkilediği, reaktif güçteki değişimin gerilimi etkilediği ilkelerine dayanmaktadır. [10]

Seethalekshmi K., yayınlamış olduğu makalede, kritik durumlar altında şebekenin kendi kendine iyileşmesi için gerekli olan güç sistemindeki yük kısıtlaması için sistemin gerilim voltajının yanı sıra frekansa da bir tehdit oluşturabilecek yeni bir şema sunmaktadır. Sistemdeki yük atma gereksinimi, hesaplanan bozulma gücünün yanı sıra sistemin voltaj dengeleme durumuna, bununla birlikte gerçek zamanlı bir zaman çizelgesine dayanarak, sistemin kontrolünden ve kontrol sisteminden elde edilebileceğini hesaplamıştır. [11]

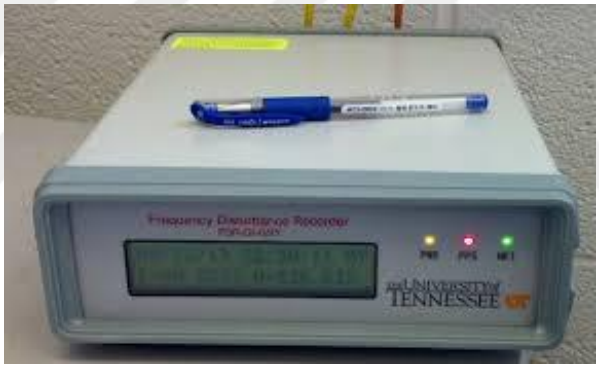
Le Xie yapmış olduğu çalışmada, senkron fazör verilerinin temel boyutsallığını incelenmiş ve azaltılmış boyutsallığı kullanarak erken olay tespiti için çevrimiçi bir uygulama geliştirmiştir. Bu uygulama, bir olay meydana geldiğinde PMU verilerinin çekirdek alanlarının değiştirilmesine dayanan bir erken olay algılama algoritmasıdır. Hem normal hem de anormal koşullar altında fazör ölçü birimi (PMU) verilerinin boyutsallığı analiz edilmiştir. Algoritmanın temel verileri, doğrusal dinamik sistem teorisi kullanılarak sağlanmıştır. Önerilen algoritmayı test etmek ve doğrulamak için hem sentetik olarak üretilen verileri hem de fazör ölçü biriminden elde edilen gerçekçi verileri kullanan sayısal simülasyonlar gerçekleştirilmiş ve bu simülasyon sonuçları değerlendirilmiştir. [12]

2. YAPILAN ÇALIŞMALAR

2.1. Kullanılan Cihaz ve Yazılımların Tanıtımı

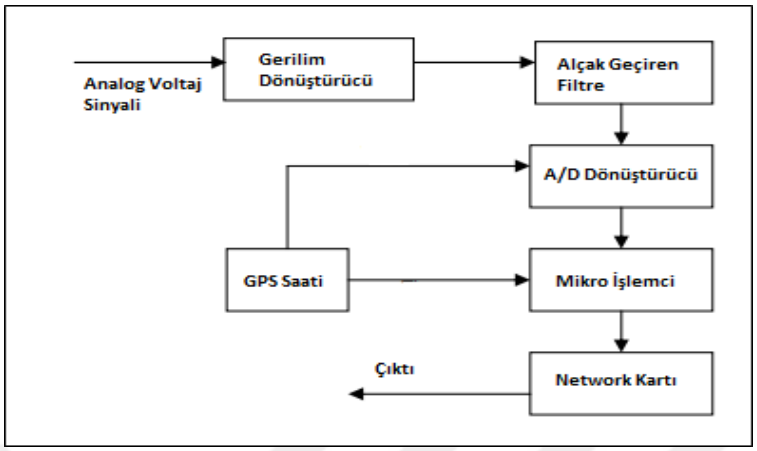
2.1.1. Frekans Bozukluğu Kaydedici (Frequency Disturbance Recorder)

Frekans Bozukluğu Kaydedici (Frequency Disturbance Recorder), GPS ile senkronize edilmiş, giriş geriliminden bağımsız çıkışı 120 V olan tek fazlı bir faz ölçüm ünitesidir.



Şekil 2.1. Frekans Bozukluğu Kaydedici

Çıkış geriliminin tipik faz ölçüm ünitesinden daha düşük tutulmasının sebebi uygulanmasının nispeten ucuz ve basit olmasıdır.



Şekil 2.2. FDR Mimarisi

FDR, analog-dijital dönüştürücü yardımıyla çıkışın voltaj sinyalini ölçekli bir biçimde çok hızlı (saniyede 1440 kere) örneklenmesiyle çalışır. Her bir numune için voltaj sinyalinin anlık faz açısı kart üzerindeki sayısal işaret işleyicide hesaplanır. Daha sonra cihaz 100ms aralıklarla faz açısını, frekansı ve gerilimin genliğini hesaplar. Bu bilgiler sunucuya iletilip burada işlenip depo edilir. Frekans ölçümleri doğruluğu ± 0.0005 Hz. olan FDR ile belirlenir. [13] Bir FDR'nin yalnızca güç çıkışı, Ethernet portu ve gökyüzünden bir görüntüye (GPS anteni için) ihtiyacı vardır. Bu tarz FDR'ler trafo, ofis hatta özel konutlar dahil olmak üzere her yerde kullanılabilir.

FDR cihazının yapılandırılabilmesi için Network Enabler Administrator yazılımı kullanılmıştır. Yazılım yerel ağda bulunan FDR cihazlarını otomatik olarak bulup listelemektedir.

2.1.2. Microsoft Visual Studio

Microsoft Visual Studio, Microsoft firması tarafından geliştirilen .Net Framework kütüphanelerini kullanan bir yazılım geliştirme ortamıdır. NET teknolojisi tüm programlar için gerekli altyapıyı tek bir çatı altında toplamaktır. Visual Studio ile .Net bünyesindeki ortak kütüphanelerin kullanılabiliyor olması sebebiyle yazılımlar Microsoft Visual Studio ortamında C#.Net yazılım diliyle gerçekleştirilmiştir.

C#, .NET ile birlikte yaratılmış bir dildir. C# dili C, C++ ve Java dillerine benzemektedir. C# dilinin çıkartılmasındaki amaç C, C++ ve Java dilini bilen insanların .NET'e geçişlerini kolaylaştırmaktır. [14]

2.1.3. Access Veritabanı

Access, Microsoft firması tarafından geliştirilmiş, ilişkisel veritabanı yönetim sistemi ile çalışan bir veritabanı oluşturma programıdır. Diğer veritabanı uygulamalarına göre kullanımının ve C#.net ile kontrolünün kolay olması sebebiyle tercih edilmiştir.

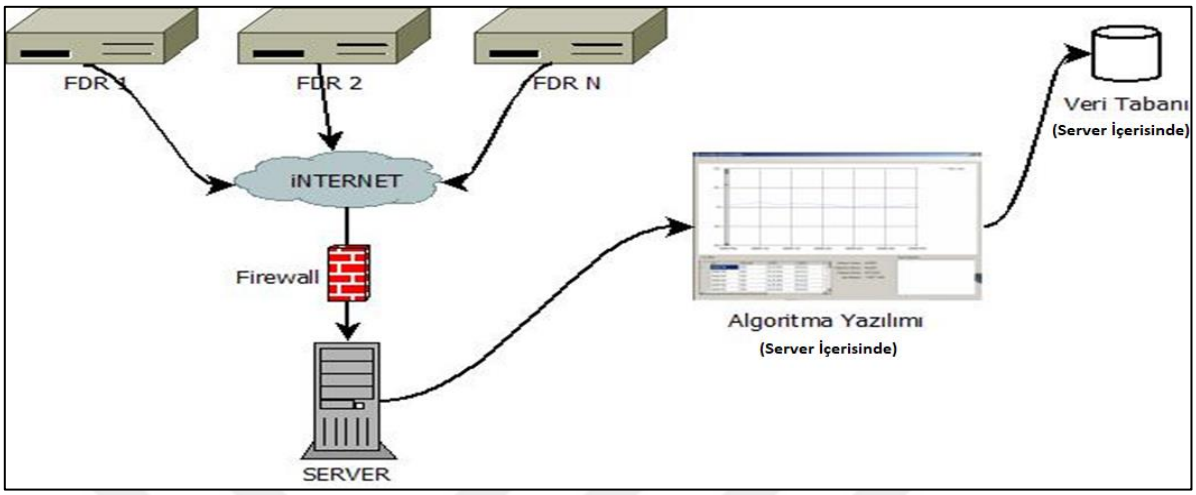
2.1.4. Matlab

MATLAB (matrix laboratory), MathWorks şirketi tarafından geliştirilmekte olan, çok paradigmalı sayısal hesaplama yazılımı ve dördüncü nesil programlama dilidir. Dördüncü nesil programlama dilleri, kullanımı çok daha kolay, daha az kod yazarak yönergeler, hazır şablonlar ve sihirbazlar sayesinde belirli ihtiyaçlarda uzmanlaşmış pratik çözümler geliştirmeye yönelik dillerdir.

Tez çalışmasında geliştirilen algoritmanın elde edilmesinde, eldeki örnek verilerin algoritma ile test edilmesinde MATLAB programı kullanılmıştır. Grafiklerde ve hesaplamalarda MATLAB uygulamasının hazır kütüphanelerinden faydalanılmıştır.

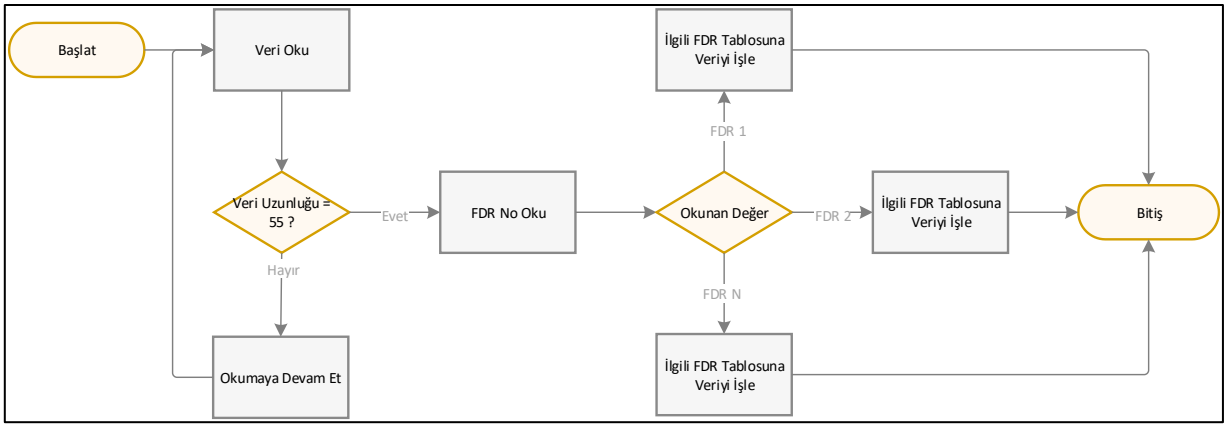
2.2. Sistem Mimarisi ve Akış Diyagramları

Enterkonnekte şebekenin farklı noktalarına konumlandırılan FDR cihazları sayesinde GPS zaman senkronlu çok sayıda frekans bilgisi elde edilmektedir. Bu bilgiler FDR cihazlarına bağlı internet LAN kablosu sayesinde, sunucunun bulunduğu ağdaki güvenlik duvarından geçtikten sonra C#.Net dilinde kodlanarak hazırlanan ve yine server içerisinde bulunan yazılım vasıtasıyla bilgiler veri tabanına kaydedilmektedir.



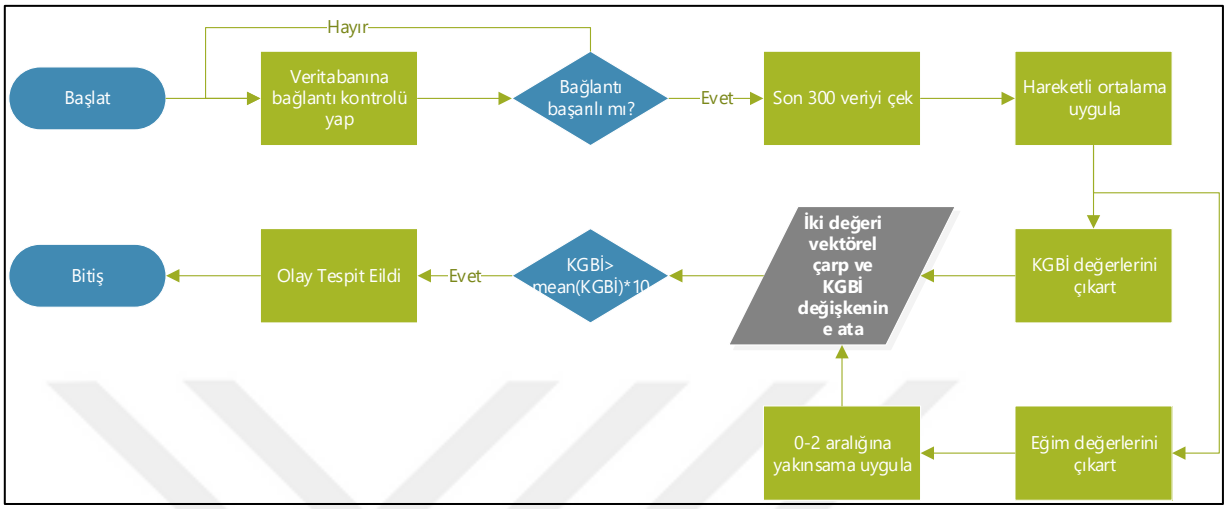
Şekil 2.3. Sistem Mimarisi

Hazırlanan yazılım verileri okurken ilgili FDR cihazına özel portu dinleyerek gelen verileri paket haline getirip kontrol ettikten sonra veri tabanında bulunan yine FDR cihazına özel tabloya kaydetmektedir.

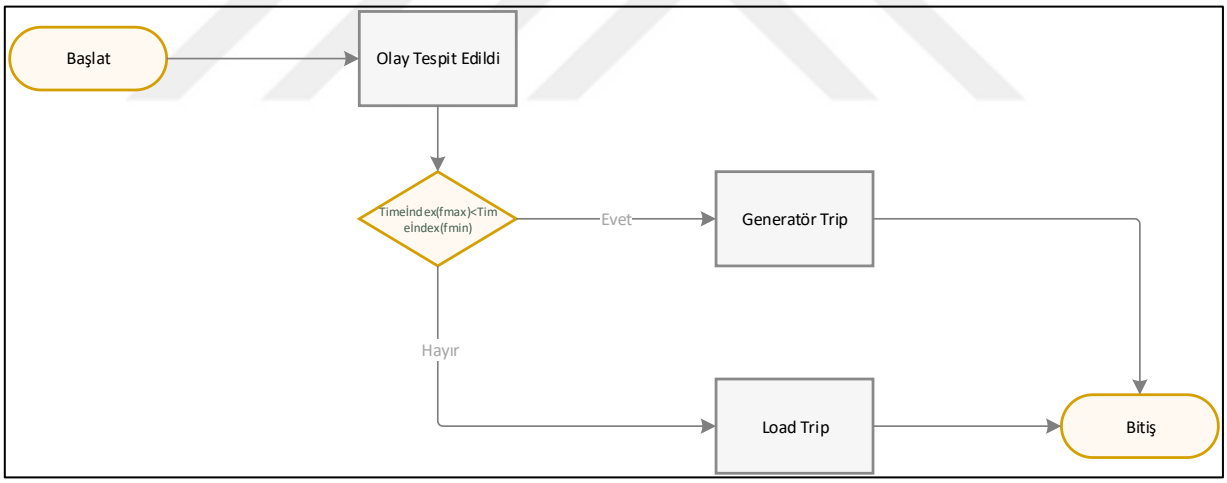


Şekil 2.4. Veri Okuma Akış Diyagramı

Frekans verileri veri tabanına işlendikten sonra aynı zamanda veri tabanından 30sn ye tekabül eden 300 adet veri çekilerek tasarlanan algoritmadan geçirilerek olay tespiti yapılmaktadır.



Şekil 2.5. Olay Yakalama Akış Diyagramı



Şekil 2.6. Olay Türü Belirleme Akış Diyagramı

2.3. Server Kurulumu ve Yapılandırılması

Geniş Bölge İzleme yazılımı ile çok fazla veri incelendiğinden ve depolandığından server olarak kullanılacak cihazın doğru seçilmesi ve yapılandırılması önemlidir. Verilerin güvenliği açısından RAID (Redundant Array of Independent Disks) kullanılmıştır. Daha iyi performans sergilemesi ve veri kaybı riskinin minimum olması sebebiyle RAID 5 kullanılmıştır.

2.3.1. RAID 0

RAID 0 kullanımında veri birçok diskin üzerine yazılır. Bu da bilgisayarın bir yerine birden fazla disk ile çalıştığı ve performansının arttığı anlamına gelir. Çünkü birden fazla sürücü veriyi okur, işler ve disk girdi / çıktısını geliştirir. En azından iki diske ihtiyaç duyar.

2.3.2. RAID 1

RAID 1 ile veri sorunsuz bir şekilde ve devamlı olarak bir diskten diğerine kopyalanır ve böylece replika ya da ikiz yapı sağlanmış olur. Eğer bir tane disk bozulursa diğeri çalışmaya devam edebilir. Bu hata toleransını kullanmanın en kolay yoludur ve aynı zamanda düşük maliyetlidir.

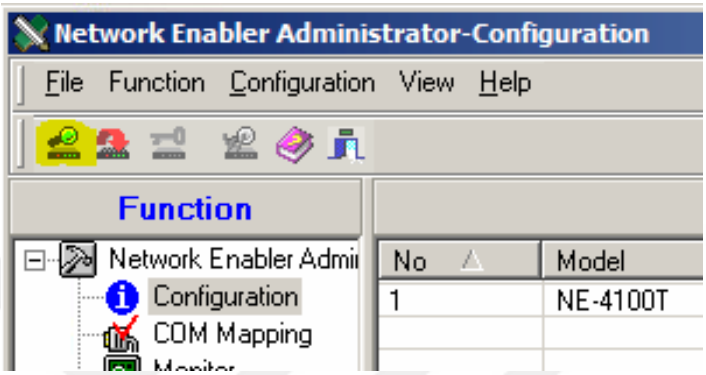
2.3.3. RAID 5

Bu RAID seviyesi RAID 1'den yani ikizlemeden daha iyi bir performans sağlar. RAID 5 ile veri, üç veya daha fazla diske bölüştürülür. Eğer bir disk hata verirse veya arıza vermeye başlarsa veri bu dağıtılmış veri ve eşlik bloğundan otomatik ve sorunsuz bir şekilde tekrar yaratılır. Temel olarak sistem, siz bozulan sürücüyü değiştirene kadar, bir diskle bile olsa çalışmaya devam eder.

2.4. FDR Cihazının Kurulumu ve Yapılandırılması

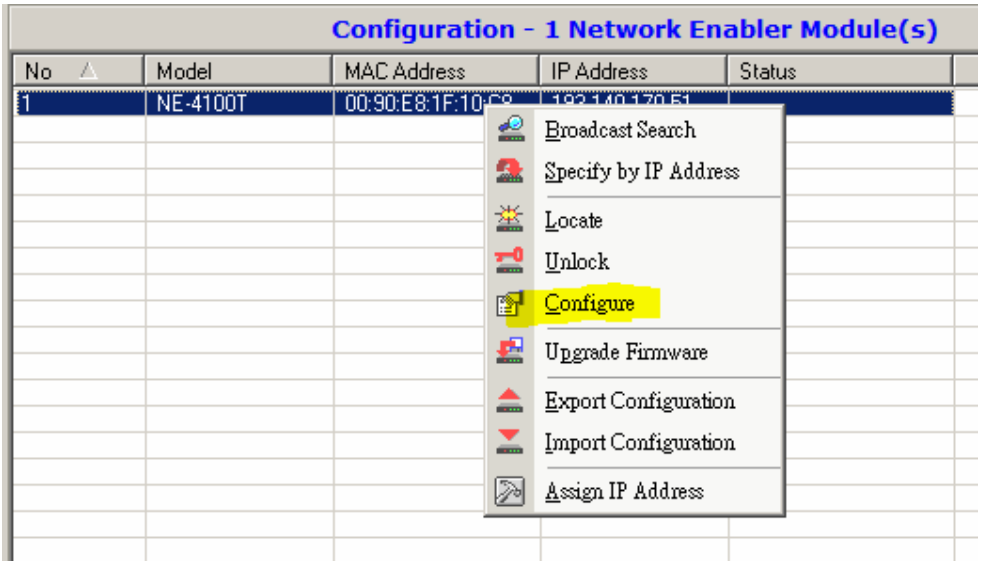
FDR cihazı bir adet GPS modeme ve internet bağlantısına ihtiyaç duymaktadır. Cihazın yapılandırılabilmesi için Network Enabler Administrator yazılımı kullanılmaktadır. Yazılım vasıtasıyla FDR cihazlarına erişim sağlanmaktadır. Erişimin sağlanabilmesi için yazılımın kurulu olduğu bilgisayar ile FDR aynı IP bloğunda yer almalıdır.

Yazılım çalıştırıldığında sol üst menüde bulunan arama butonuna basılarak aynı IP bloğundaki FDR cihazları görüntülenir.



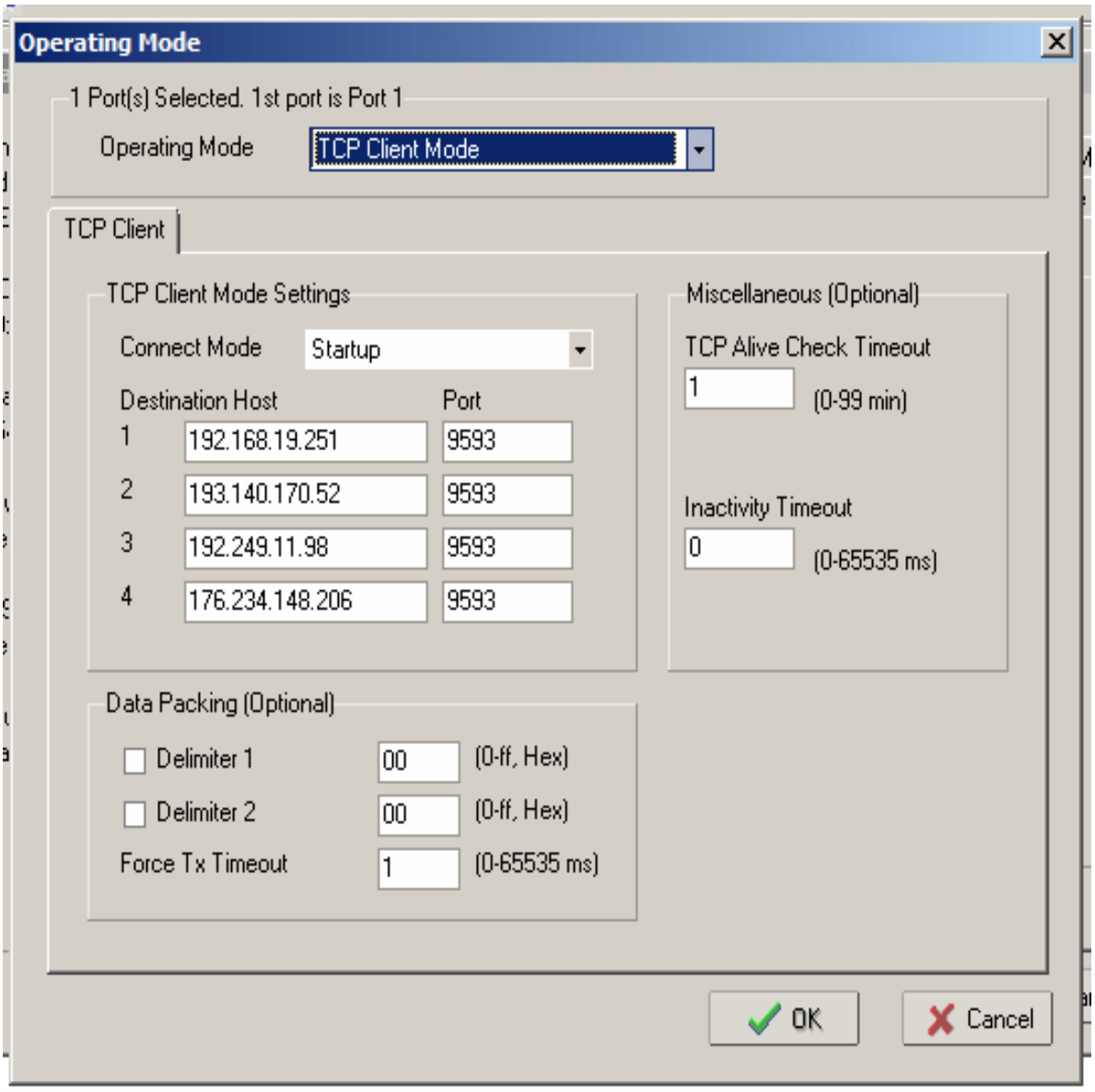
Şekil 2.7. FDR Tarama

Tarama sonrası bulunan FDR cihazları datagrid üzerinde listelenmektedir. İşlem yapılmak istenen cihaz üzerinde sağ tıklanarak configure seçeneği seçilir.



Şekil 2.8. FDR Liste Penceresi

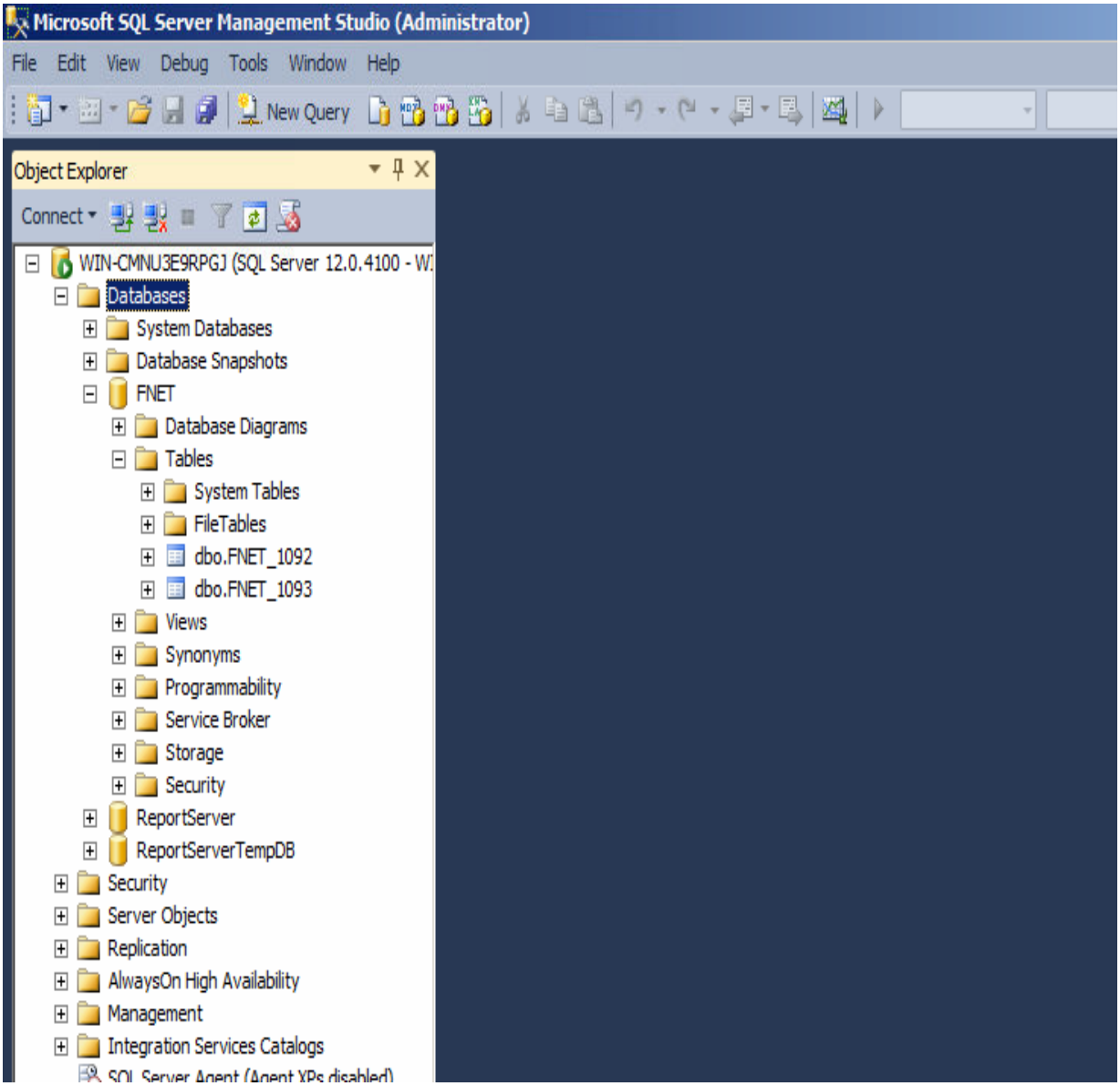
Açılan ayar penceresinde birçok sekme bulunmaktadır. Verinin hangi IP ve portlara gönderileceğine yönelik ayarlamalar ***Operating Mode*** sekmesi altında bulunmaktadır. TCP üzerinden bilgi akışının sağlanması istendiğinden ***TCP Client Mode*** seçilerek ***Destination Host*** ve ***Port*** bilgileri ilgili yerlere yazılmıştır.



Şekil 2.9. TCP Client Ayarları

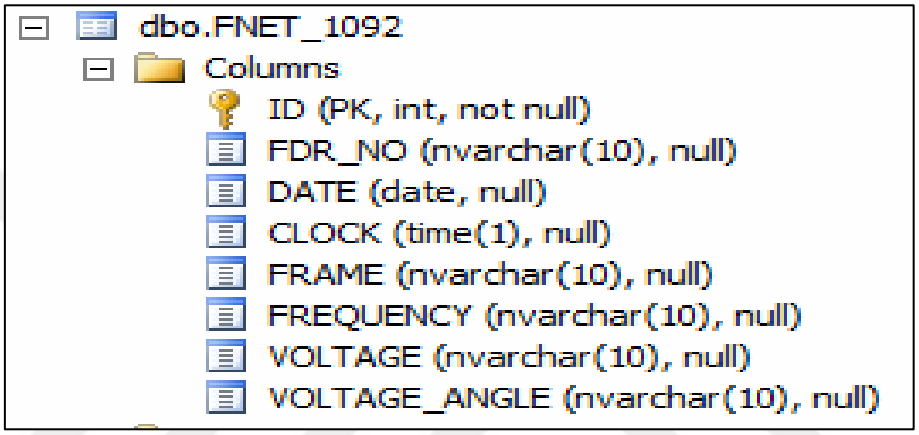
2.5. Veritabanı Oluşturma

FDR cihazından gelen verilerin hem anlık hem de daha sonrasında incelenebilmesi için anlamlı bir şekilde depolanması gerekmektedir. Veritabanı olarak Microsoft SQL Server Management Studio yazılımı kullanılmıştır.



Şekil 2.10. Microsoft SQL Server Management Studio

Yazılım içerisinde FNET adında Veritabanı oluşturulmuştur. Her bir FDR cihazı için verilerin karışmaması adına ayrı ayrı tablolar oluşturularak her FDR a ait bilgiler ilgili tablolara yazılarak kayıt altına alınmıştır.



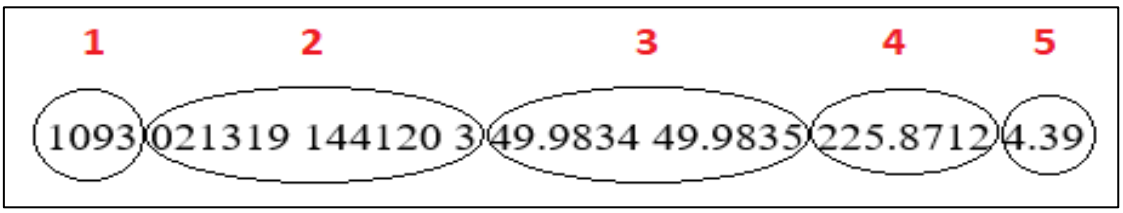
Şekil 2.11. Tablo Sütunları

Her FDR cihazından gelen ve düzenlenen veriler yukarıdaki tabloda da gözüken ilgili sütunlara kaydedilmektedir. Verilerin karışmaması ve düzenli bir sıra ile kaydedilmesi adına benzersiz arttırılmış sayı üreten ID sütunu da kullanılmıştır.

2.6. Veri Depolama İşlemleri

Elektrik şebekesinde meydana gelen olayların anlık ve geriye dönük olarak incelenebilmesi için frekans dinamiklerinin düzgün ve etkin bir şekilde depolanması ve sınıflandırılması gerekmektedir. Farklı cihaz ve noktalardan gelecek olan bilgi akışının birbiri ile karışmadan ve bilgi kaybı yaşanmadan depolanabilmesi için izlenen yol aşağıda detaylarıyla verilmiştir.

FDR cihazı, saniyede 10 adet veri göndermektedir. Bu veri; FDR cihazının numarası, tarih, saat, frekans, gerilim ve faz açısı bilgilerini içermektedir.



Şekil 2.12. FDR Verisi

- (1) FDR cihazının numarası : 1093
- (2) Veriye ait tarih saat bilgisi: 13.02.2019 14:41:20:03 UTC
- (3) Frekans Bilgisi : 49.9834 Hz
- (4) Gerilim : 225.8712 V
- (5) Faz Açısı : 4.39 º

2.7. Veri İşleme

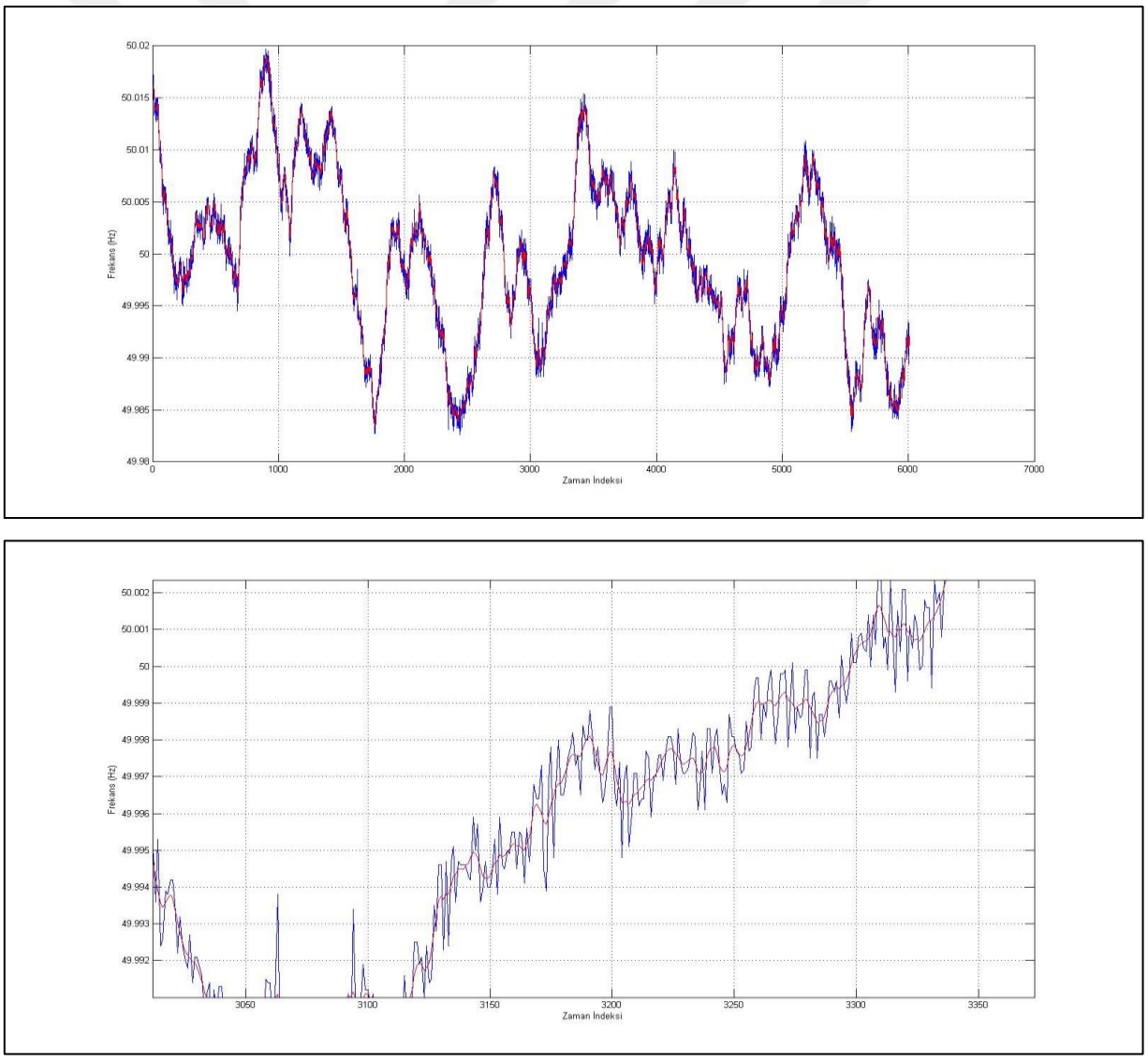
Filtreler, verilerdeki yüksek frekanslı dalgalanmaları gideren veya belirli bir frekansın periyodik trendlerini verilerden kaldırabilen veri işleme teknikleridir. FDR cihazından gelen bilgilerin doğrudan kullanılması, olayların tespitinin doğruluğunu düşürecektir. Daha etkin ve doğru sonuçların elde edilebilmesi için verilerin daha işlevsel hale getirilmesi gerekmektedir. Özellikle frekans bilgisi anlık olarak sürekli dalgalanan ve bünyesinde gürültü bulunduran bir sinyaldir. Frekans bilgisi üzerindeki bu gürültülerin etkisini azaltmak için hareketli ortalamalar yöntemi ile veri filtrelenmiştir.

2.7.1. Üçgensel Ortalama Filtresi

Üçgensel ortalama filtresi basit bir yapıya sahiptir ve eğilimin genel yönünü gösterir. FIFO yapısıyla değerler, filtre boyutuna göre üçgen form oluşturacak katsayılar ile çarpılır ve bu katsayıların toplamına bölünür.

$$Y_t = \frac{Y_{t-2}+2*Y_{t-1}+3*Y_t+2*Y_{t+1}+Y_{t+2}}{k} \quad (1)$$

Formüldeki k katsayısı oluşturulan üçgen şeklinin katsayılarının toplamıdır. Bu katsayı arttıkça filtreleme miktarı artacaktır. Katsayının çok büyük seçilmesi eldeki verinin bozulmasına, çok küçük seçilmesi de yeteri kadar filtreleme işleminin yapılamamasına sebep olacaktır. Bu nedenle eldeki verinin gürültü seviyesine göre optimum bir k katsayısı belirlenmelidir.



Şekil 2.13. Üçgensel Ortalama Filtresi

Yukarıdaki Şekil 2.9'da de görüleceği üzere üçgensel ortalama filtresi uygulanan veri, kullanılan katsayı oranında filtrelenerek daha temiz ve yorumlanabilir hale gelmektedir.

Eldeki verilerin test edilebilmesi için filtreleme işlemi Matlab yazılımı vasıtasıyla kodlanmıştır. Üçgensel ortalama filtresi yöntemi Matlab ortamında fonksiyon olarak hazırlanmıştır.

```
%%%%%%%%%%%%

function [ df ] = ucgenselOrtalama( f )

    df(1)=f(1);
    df(2)=f(2);
    df(3)=f(3);
    df(size(f,1)-2)=f(size(f,1)-2);
    df(size(f,1)-1)=f(size(f,1)-1);
    df(size(f,1))=f(size(f,1));

    for i=4:size(f,1)-3

        df(i)= (f(i-3)+2*f(i-2)+3*f(i-
1)+4*f(i)+3*f(i+1)+2*f(i+2)+f(i+3))/16;

    end

df=df';

end
%%%%%%%%%%%%
```

Hazırlanan fonksiyonda 1 adet giriş değeri ve 1 adet çıkış değeri bulunmaktadır.

Tablo 2.1. Üçgensel Ortalama Filtresi Değişkenleri

Veri Adı	**Tipi**	**Veri Türü**	**Açıklama**
f	Girdi	Array	Filtrelenecek veri
df	Çıktı	Array	Fitreleme işlemi sonrası oluşan dizi

2.7.2. Eğim Hesaplama Algoritması

Hareketli ortalama yöntemiyle filtrelenen veri üzerinde eğim hesabı yapılarak, frekansın anlık tepkilerinin şiddeti ölçülmüştür. Her iki örnekleme arası oluşturulacak üçgen için eğim, karşı ve komşu kenarların birbirine oranı olacaktır. Filtrelenen ardışık iki frekans verisi arasındaki eğim hesaplanırken komşu kenar 1 olacağından iki frekans verisi arasındaki fark eğim bilgisini verecektir.

$$eğim = f_n - f_{n-1} \tag{2}$$

```
%%%%%%%%%%%%
function [ output_args ] = egim( input_args )
for i=2:size(input_args,1)
    output_args(i-1)=abs((input_args(i)-input_args(i-1)));
end
    output_args=output_args';
end
%%%%%%%%%%%%
```

Tablo 2.2. Eğim Hesaplama Fonksiyonu Değişkenleri

Veri Adı	Tipi	Veri Türü	Açıklama
input_args	Girdi	Array	Giriş verisi
output_args	Çıktı	Array	Eğim çıktısı

2.7.3. Normalizasyon Algoritması

Olay tespitinde esas teşkil eden kgbi katsayısının frekansın ani tepki şiddetinin etkisi altında olmasını sağlamak amacıyla eğim algoritmasıyla hesaplanan dizi ile çarpılacaktır. Ancak

bu etkinin, eğim dizisinin tüm hareketlerinden etkilenmesi, gürültülerin kgbi katsayısını anlamsız hale getirmesine neden olacaktır. Bu sebeple eğim dizisinin belirli kurallar çerçevesinde normalizasyonu gerekmektedir.

Normalizasyon algoritması, bir dizinin belirlenen sayı etrafında toplanması ve sınırlandırılması için hazırlanmış algoritmadır. Matlab ortamında hazırlanan fonksiyonda ki k girdisi, normalizasyonu yapılacak olan dizi elemanlarının etki sınırını belirleyen sınır katsayıdır. Dizi içerisindeki değerler bu sayıya normalize edilerek maksimum k değerini alması sağlanmaktadır.

$$f_n = \frac{kxf_n}{f_{maks}} \qquad (3)$$

Normalizasyon algoritması ile, bir dizinin bir başka dizi üzerindeki etkisi sınırlandırılacak şekilde, dizinin en büyük değeri sınır değerine eşitlenerek diğer değerlerin büyüklükleri oranında sınır değere yaklaşmasıyla yeni bir dizi elde edilir. Bu dizi, dizi çarpım algoritması ile etkisini bir diğer dizi üzerinde sınırlandırılmış olarak göstermektedir.

```
%%%%%%%%%%%%
function [ output_args ] = yakinsa( input_args,k )

maks=max(input_args(100:end));
output_args=input_args/(maks/k);
output_args=output_args';
end
%%%%%%%%%%%%
```

Tablo 2.3. Yakınsama Fonksiyonu Değişkenleri

Veri Adı	**Tipi**	**Veri Türü**	**Açıklama**
input_args	Girdi	Array	Veri dizisi
k	Girdi	İnteger	Yakınsanacak değer
output_args	Çıktı	Array	Sonuç dizisi

2.7.4. Kgbi Referans Değerinin Belirlenmesi

Kgbi, belirlemiş olduğumuz çerçeve içerisindeki frekans değerlerinden elde edilen ve olay tespitinde kullanılan referans değeridir. Kgbi katsayısı çerçevedeki frekans değişiminin ortalama üzerindeki etkisini göstermektedir.

$$K_{gbi} = (f_{maks} - f_{min}) \times f_{ortalama} \quad (4)$$

Örnek veriler üzerinde test yapılabilmesi adına Matlab ortamında Kgbi referans değerini bulan fonksiyon hazırlanmıştır. Yazılan fonksiyonda çerçeve boyutu değişken olarak tanımlanmıştır. Bu sayede çerçeve boyutu için de farklı değerler test edilerek en uygun değer elde edilmeye çalışılmıştır.

```
%%%%%%%%%%%
function [ output_args ] = kgbi( input_args,k )
for i=(k+1):size(input_args,1)
    maks=max(input_args(i-k:i));
    minimum=min(input_args(i-k:i));
    ort=mean(input_args(i-k:i));
    output_args(i-k)=(maks-minimum)/ort*10000;
end
```

```
output_args=output_args';

%%%%%%%%%%%%
```

Tablo 2.4. Kgbi Referans Katsayısı Değişkenleri

Veri Adı	**Tipi**	**Veri Türü**	**Açıklama**
input_args	Girdi	Array	Veri dizisi
k	Girdi	İnteger	Çerçeve boyutu
output_args	Çıktı	Array	Elde edilen kgbi değerleri

2.7.5. Dizi Çarpımı

Kgbi katsayısının, daha öncesinde filtrelenmiş frekans verisinin anlık tepkilerini ölçmek amacıyla hesaplanan eğim dizisiyle çarpılması için hazırlanan algoritmadır. Kgbi katsayısı olay tespiti öncesi son şeklini bu işlem sonrası almaktadır. Kgbi katsayısının, eğim dizisi içerisinde bulunan anlamsız değişkenlerden etkilenmemesi adına, eğim dizisi belirlenen değere yakınsanarak kullanılmıştır.

```
%%%%%%%%%%%%
function [ output_args ] = carp( input_args1,input_args2 )


    for i=1:(min(size(input_args1,1),size(input_args1,1)))

        output_args(i)=input_args1(i)*input_args2(i);

    end

end
%%%%%%%%%%%%
```

Tablo 2.5. Dizi Çarpım Fonksiyonu Değişkenleri

Veri Adı	**Tipi**	**Veri Türü**	**Açıklama**
input_args1	Girdi	Array	Veri dizisi 1
input_args2	Girdi	İnteger	Veri dizisi 2
output_args	Çıktı	Array	Çarpım işlemi sonucu

2.7.6. Olay Yakalama ve Olay Türü Belirleme

Olayın tespitinde kullanılan ana değişkenimiz FDR cihazlarından okunarak veri tabanına işlenen şebekeye ait frekans bilgisidir. Frekans enterkonnekte şebekede olan tüm olaylara tepki veren bir bileşendir. Çeşitli filtreler ve matematiksel algoritmalardan geçtikten sonra son halini Kgbi katsayısı olarak almaktadır.

Hesaplanan Kgbi referans katsayıları üzerinde belirlenen büyüklükte çerçeve hareket ettirilmektedir. Bu çerçeve içerisindeki son değer, o andaki çerçevenin ortalamasının belli bir katına ulaştığında olay tespit edilir.

Generatör devreden çıktığı zaman frekans değerinde düşme meydana gelmektedir. Yük atmasında ise tam tersi yaşanır ve frekans değerinde yükselme meydana gelir. Olayın yakalanmasından sonra türünün tespiti için bu bilgiler doğrultusunda, çerçevenin içerisindeki frekans değerlerinin ortalaması ile maksimum ve minimum değerlerinin kıyaslaması yapılır.

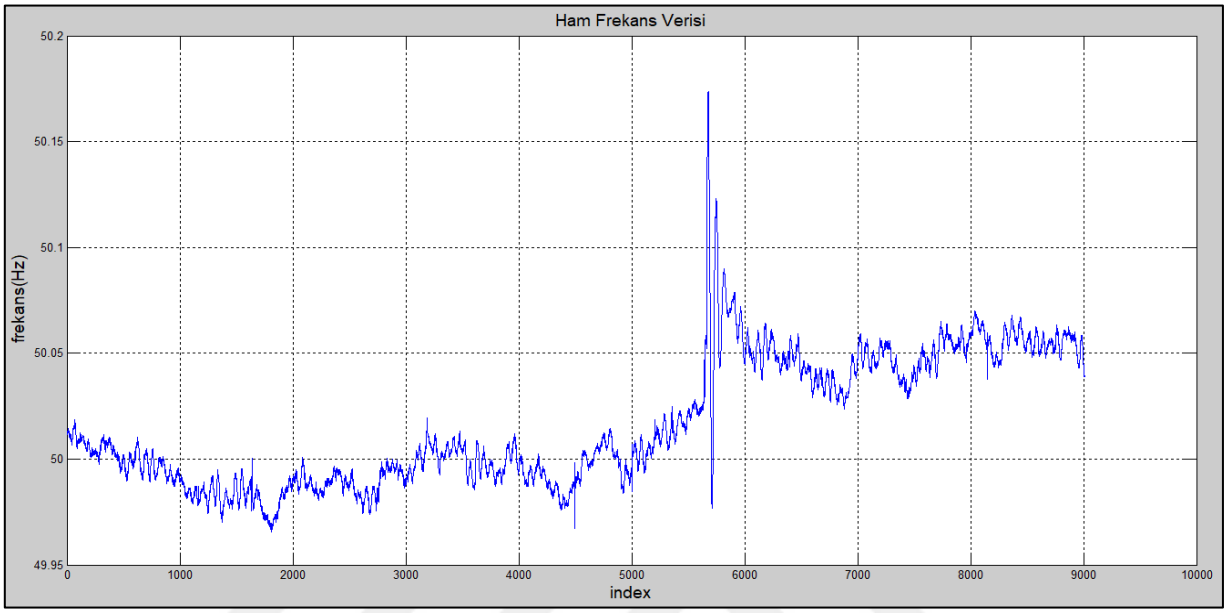
Generatör devreden çıktığı zaman frekans ani olarak düşer. Bu sebeple Kgbi katsayısı belirlenen sınırın üstüne çıktığı andaki çerçevede maksimum değer, ortalama değerden ileride olur. Yük atması durumunda ise Kgbi katsayısı belirlenen sınırın üstüne çıktığı andaki çerçevede minimum değer, ortalama değerden ileride olur.

2.8. Algoritmanın Test Edilmesi

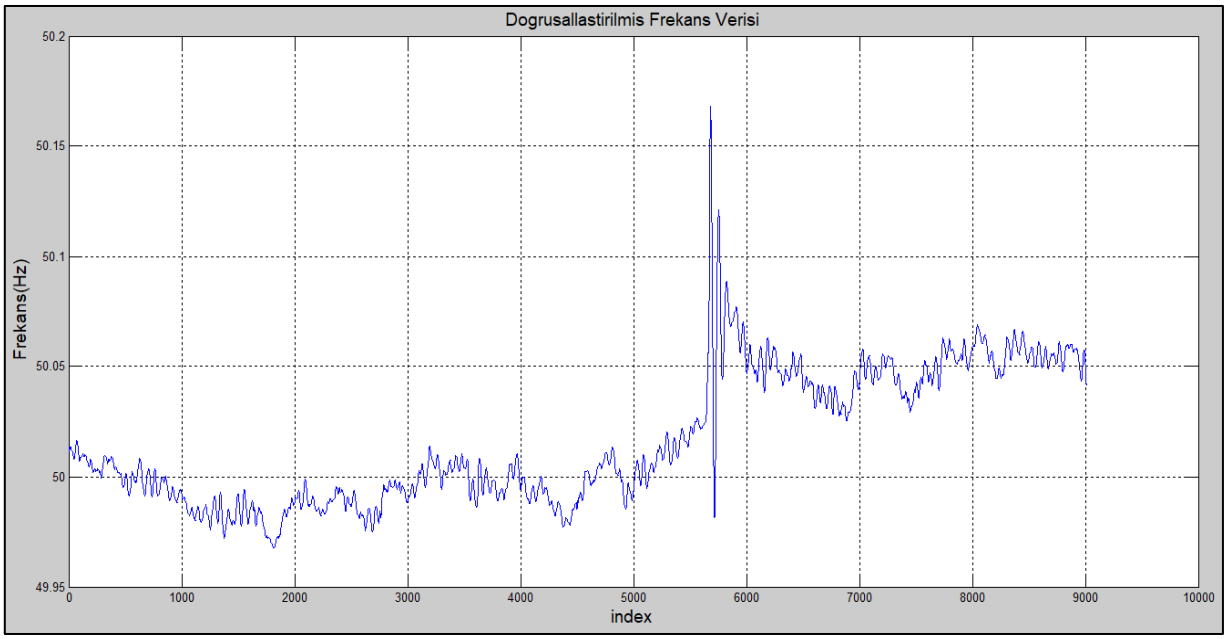
Geliştirilen olay yakalama algoritmasının test edilmesi için eldeki örnek olay verileri kullanılmıştır. Verilerin okunması, filtrelenmesi ve kgbi referans noktalarının elde edilmesi amacıyla Matlab ortamında test adında scipt hazırlanmıştır.

```
% Workspaceteki tüm verilerin silinmesi
clear all;

% Command Window ekranının temizlenmesi
clc;

%Test verisinin okunması
veriseti= load('test.txt');

% okunan veriden frekans verisinin çekilmesi
f=veriseti(:,2);

% Üçgensel Ortalama Yöntemi ile verinin filtrelenmesi
df=ucgenselOrtalama(f);

% Egim verisinin elde edilmesi
m=egim(df);

% Egim verisinin normalizasyonu
m=normalizasyon(m,2);

% filtrelenmiş veriden kgbi değerinin elde edilmesi
k=kgbi(df,300);

% Kgbi referans değerinin, eğim verisi ile güçlendirilmesi
k=carp(k,m);

% Kgbi referans değerine göre olay tespiti
olayYakalama(k);

% Tespit edilen olayın türünün belirlenmesi
olayTuruBelirle();



% Frekans verisinin ekrana çizdirilmesi
plot(f);

grid;
```

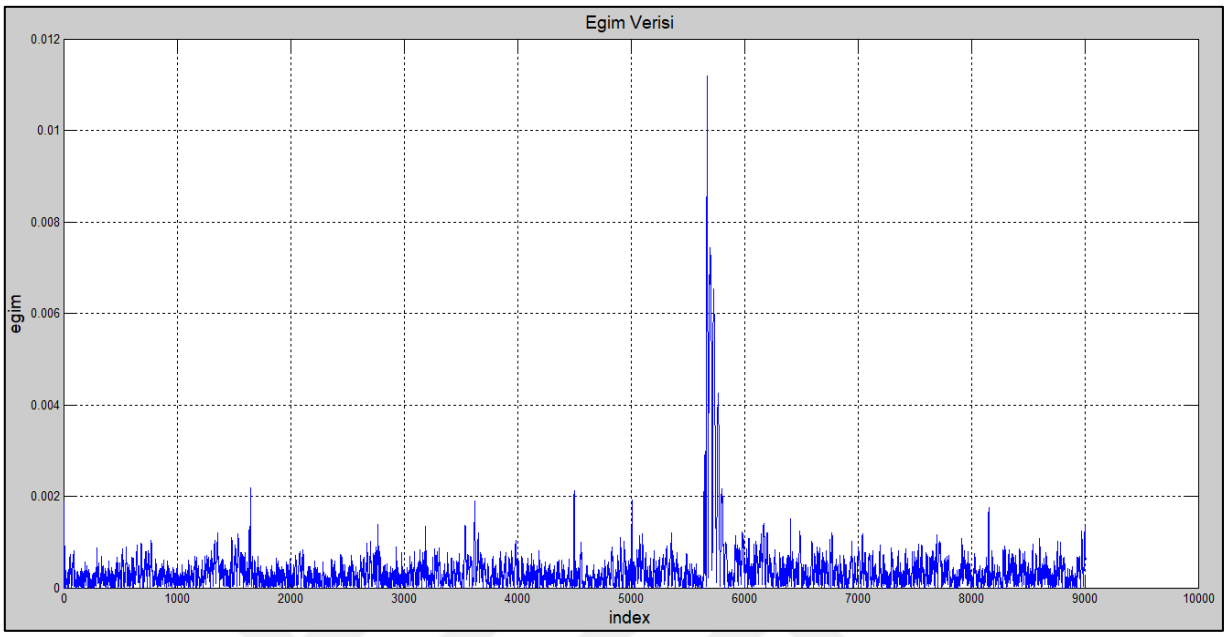
2.8.1. 25.02.2012 Tarihli 1468MW Load Trip



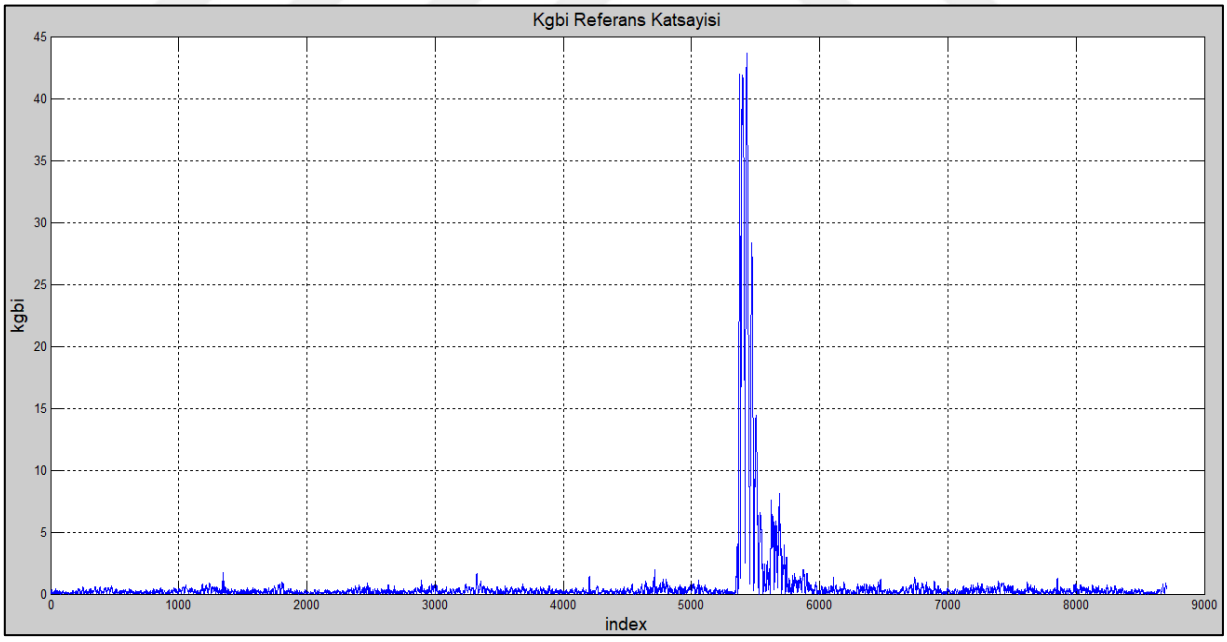
Şekil 2.14. Ham Frekans Verisi



Şekil 2.15. Filtrelenmiş Frekans Verisi

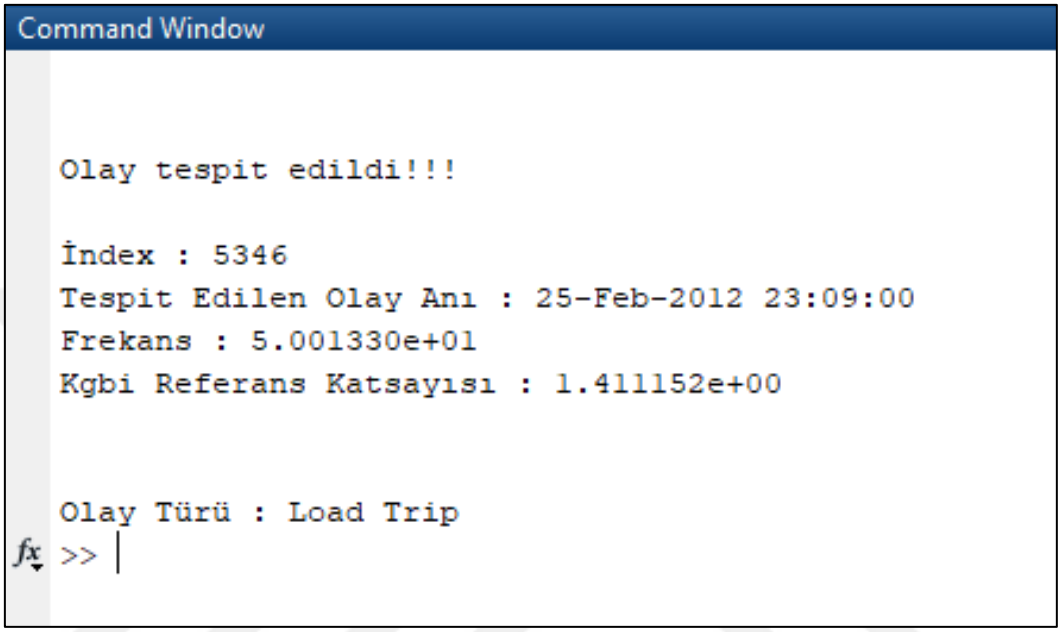


Şekil 2.16. Eğim Verisi



Şekil 2.17. Kgbi Referans Katsayısı

Algoritmanın tespit ettiği olaya ilişkin detaylar şu şekildedir;

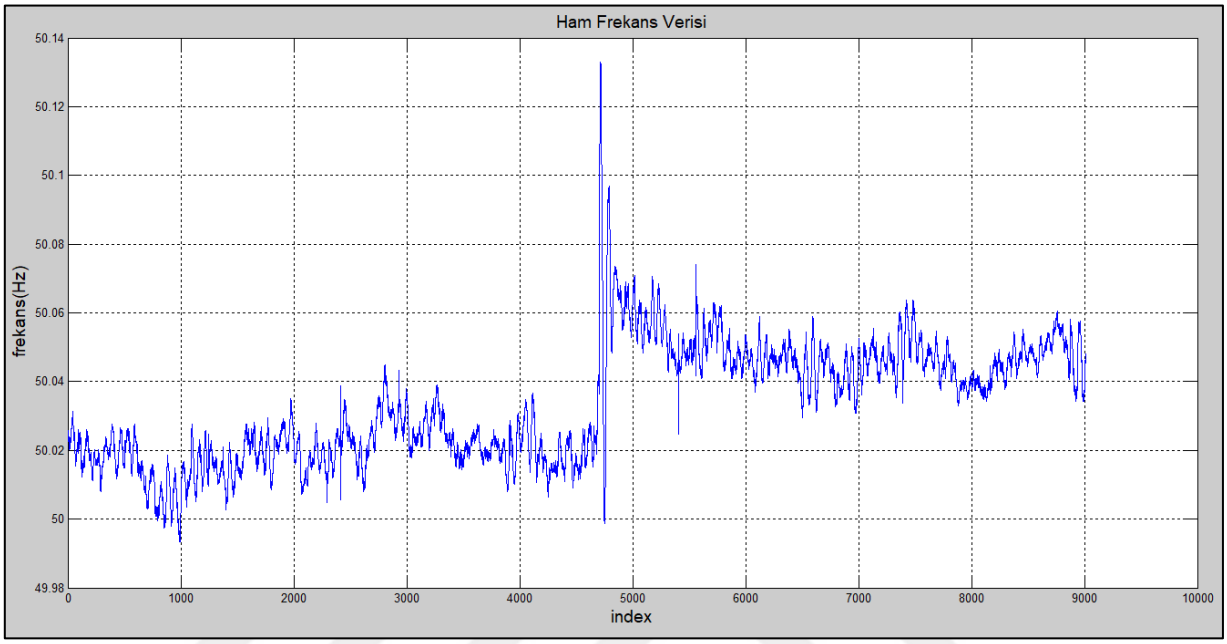


Şekil 2.18. 25.02.2012 Tarihli Test Edilen Veri Sonucu

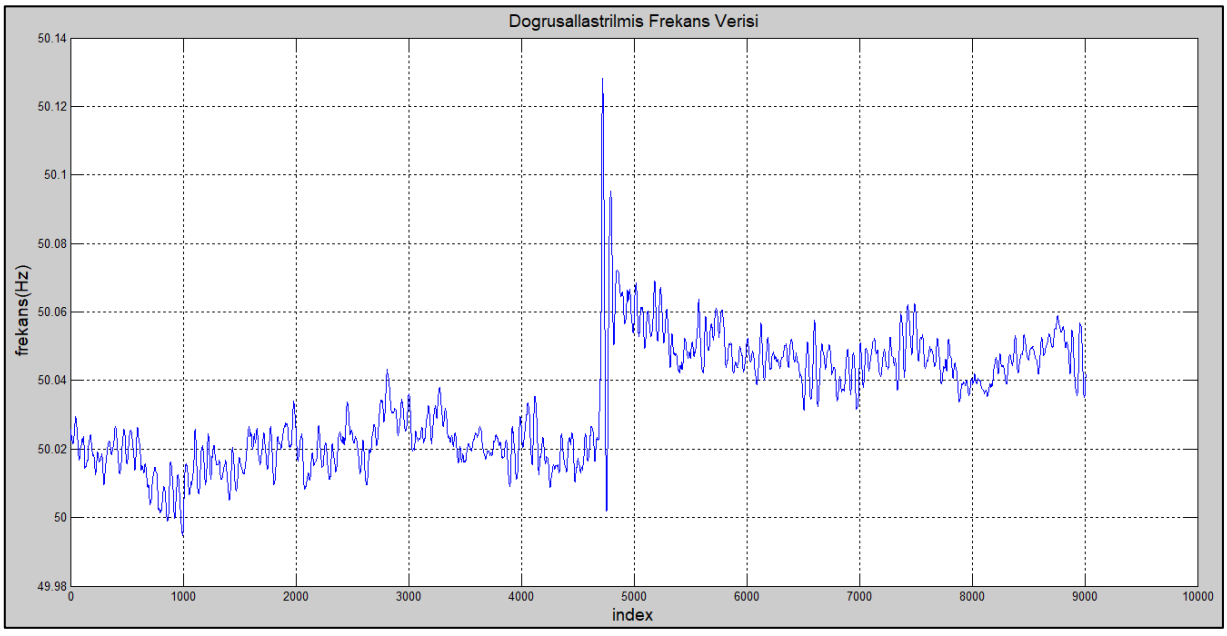
İndex	5346
Tespit Edilen Olay Anı	25-Feb-2012 23:09:00
Frekans	50.0983 Hz
Kgbi Referans Değeri	1.411152
Olayın Gerçekleşme Anı	25-Feb-2012 23:09:00
Nominal Frekans	50 Hz

25.02.2012 tarihinde meydana gelen 1468 MW Load Trip olayını, algoritma %100 başarı ve 0 sapma ile tespit etmiştir.

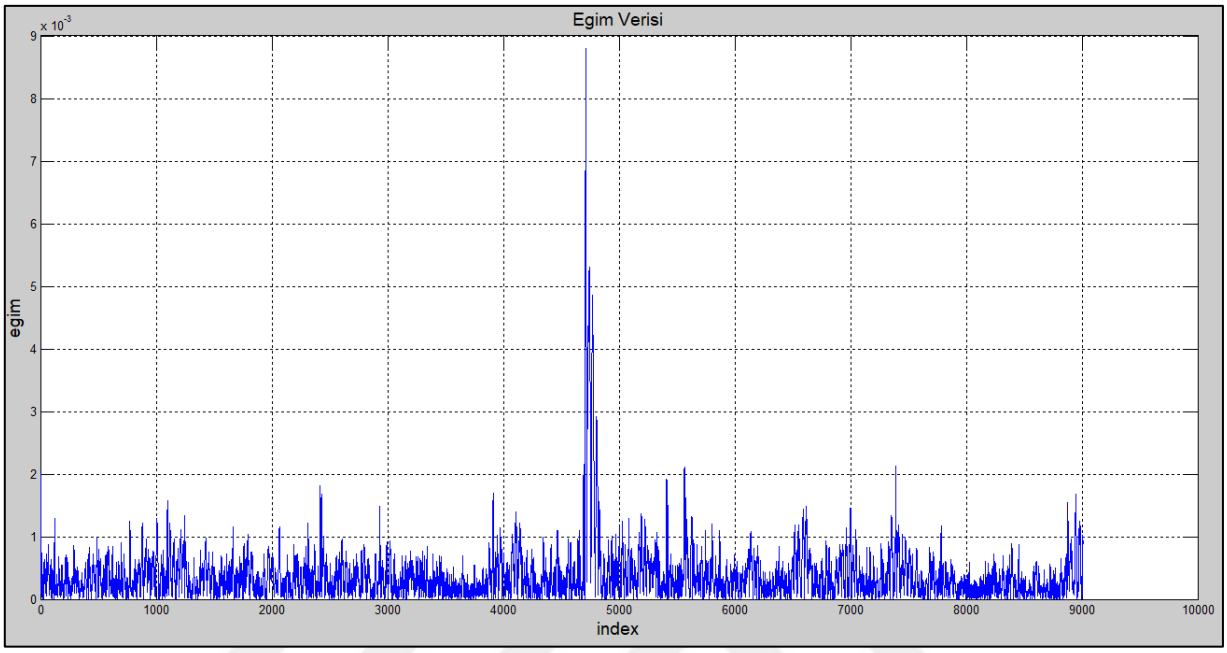
2.8.2. 26.02.2012 Tarihli 1480MW Load Trip



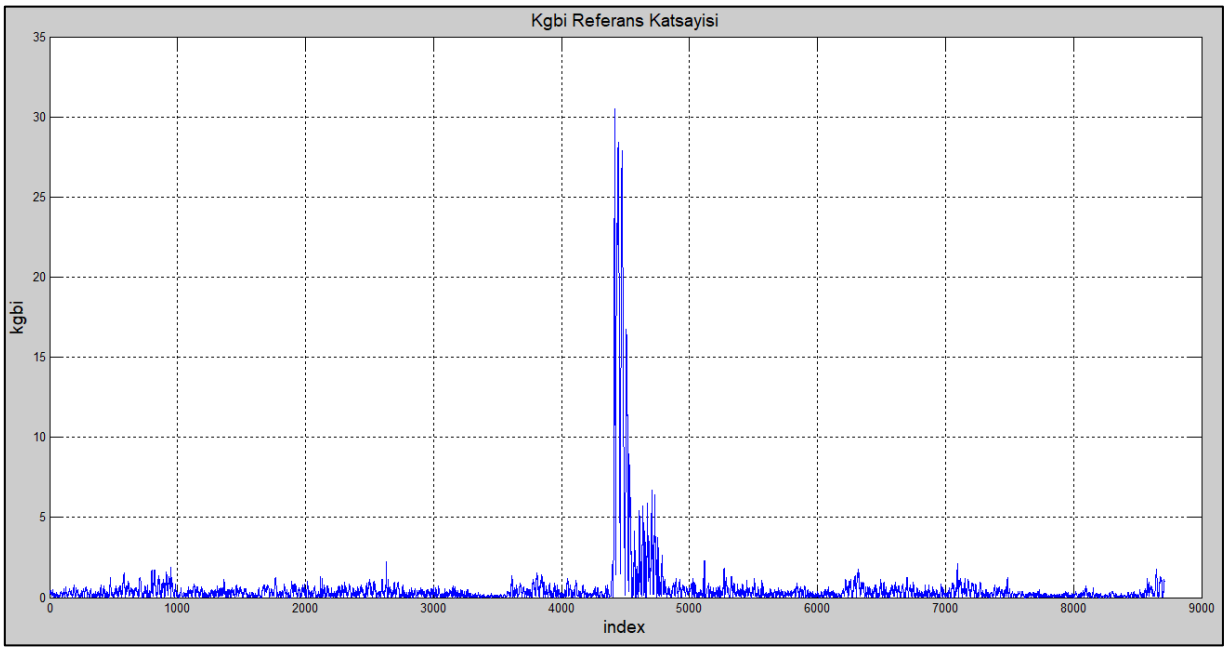
Şekil 2.19. Ham Frekans Verisi



Şekil 2.20. Filtrelenmiş Frekans Verisi

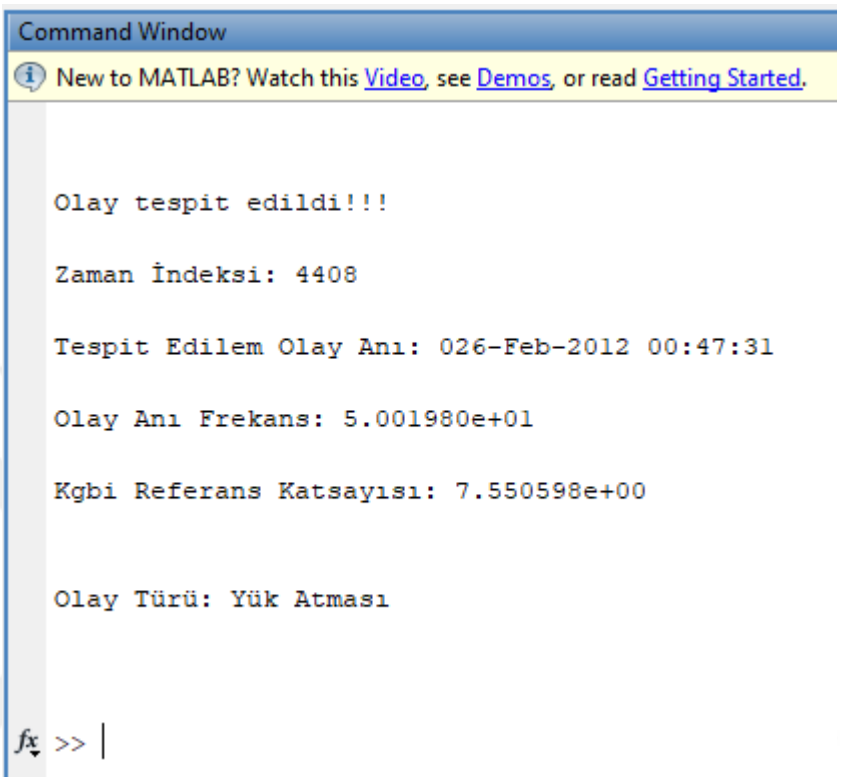


Şekil 2.21. Eğim Verisi



Şekil 2.22. Kgbi Referans Katsayısı

Algoritmanın tespit ettiği olaya ilişkin detaylar şu şekildedir;

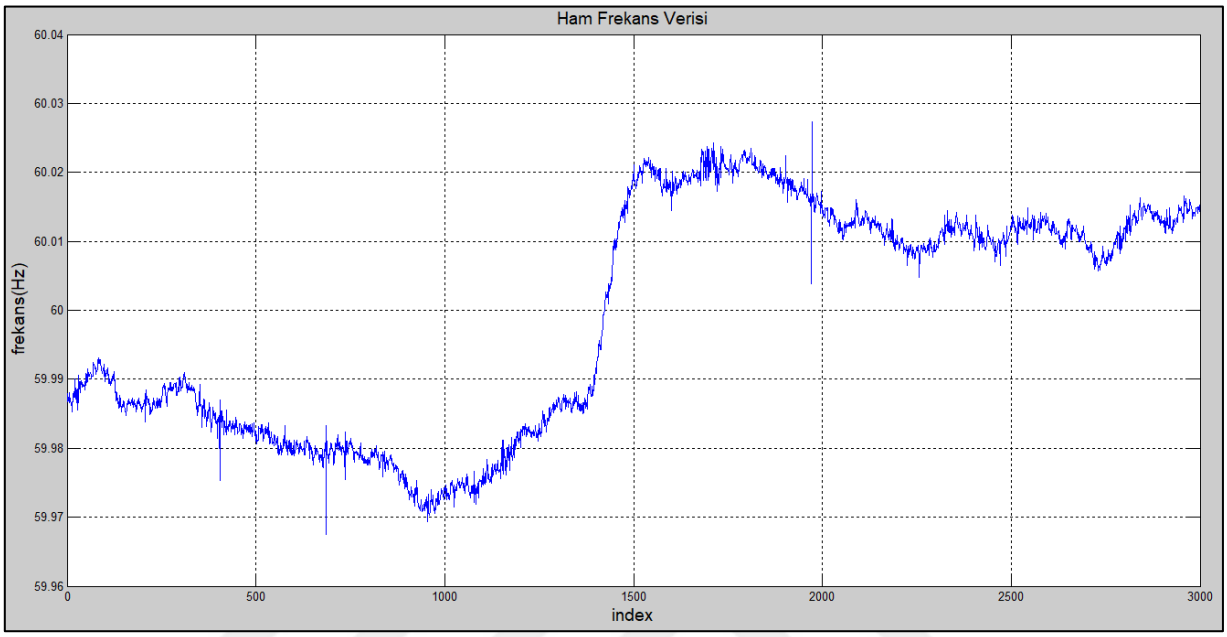


Şekil 2.23. 26.02.2012 Tarihli Test Edilen Veri Sonucu

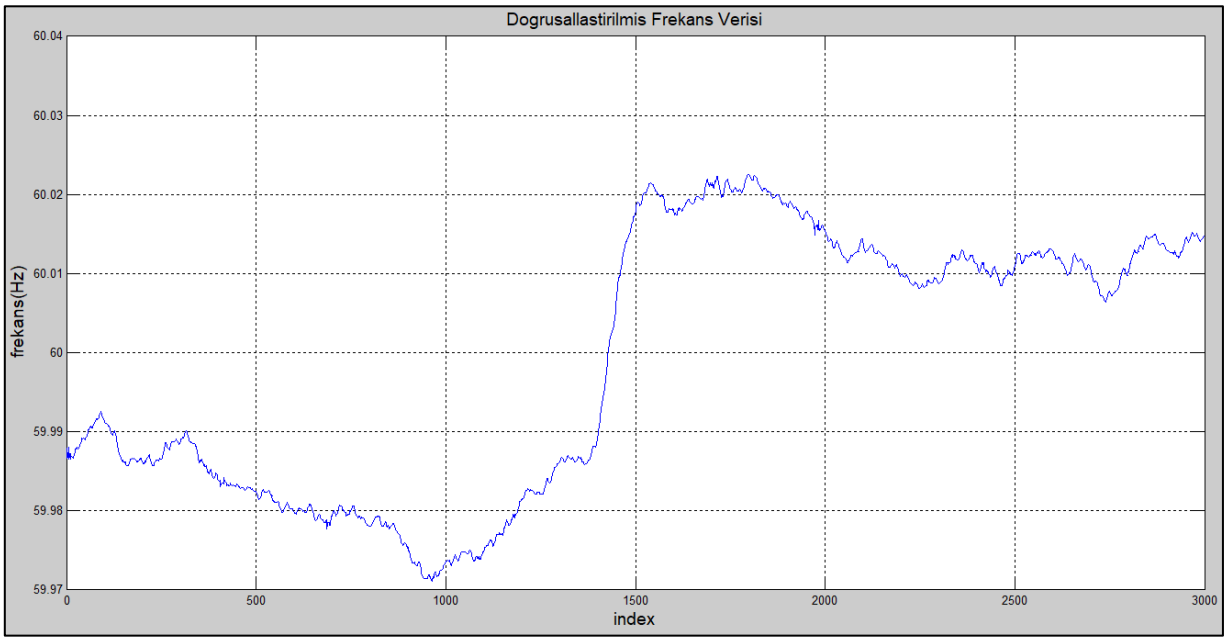
İndex	4408
Tespit Edilen Olay Anı	26-Feb-2012 00:47:31
Frekans	50.0198 Hz
Kgbi Referans Değeri	7.550598
Olayın Gerçekleşme Anı	26-Feb-2012 00:47:30
Nominal Frekans	50 Hz

26.02.2012 tarihinde meydana gelen 1480 MW Load Trip olayını, algoritma %100 başarı ve 1 sn sapma ile tespit etmiştir.

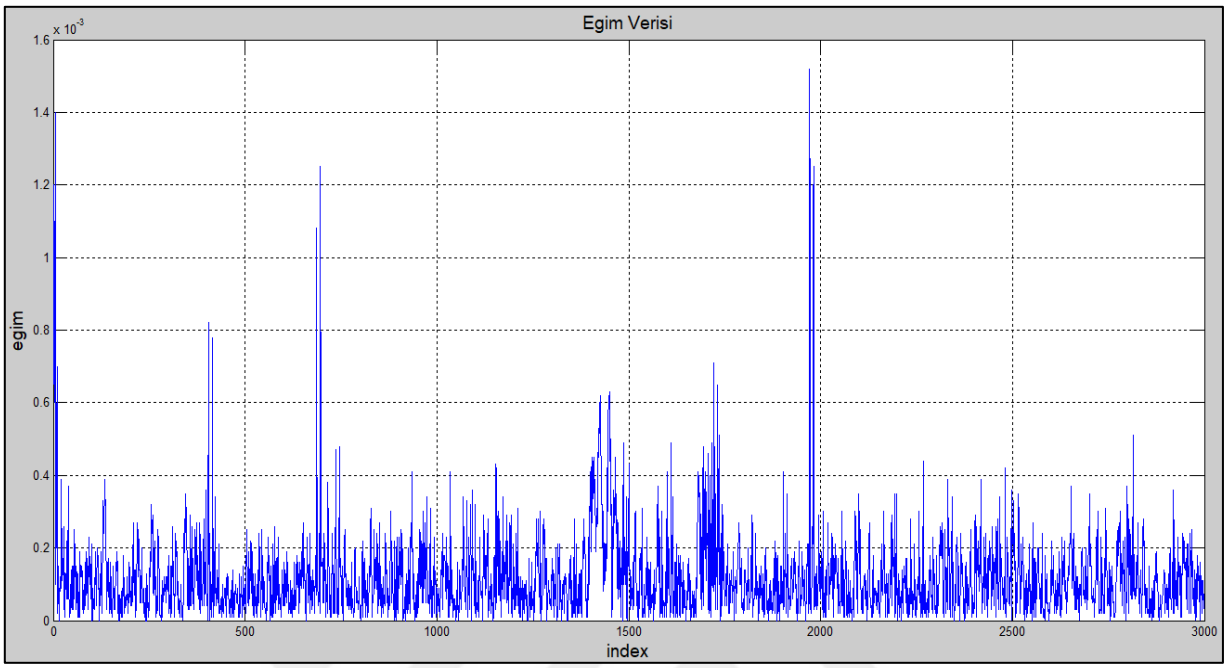
2.8.3. 09.01.2018 Tarihli Load Trip



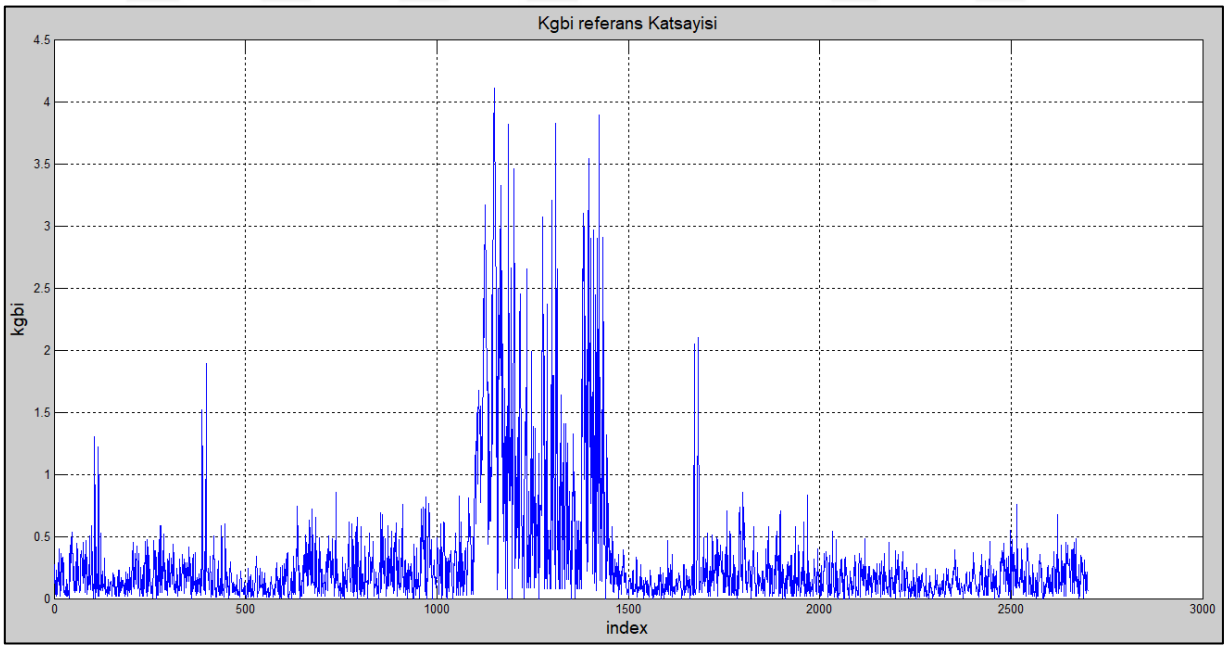
Şekil 2.24. Ham Frekans Verisi



Şekil 2.25. Filtrelenmiş Frekans Verisi



Şekil 2.26. Eğim Verisi



Şekil 2.27. Kgbi Referans Katsayısı

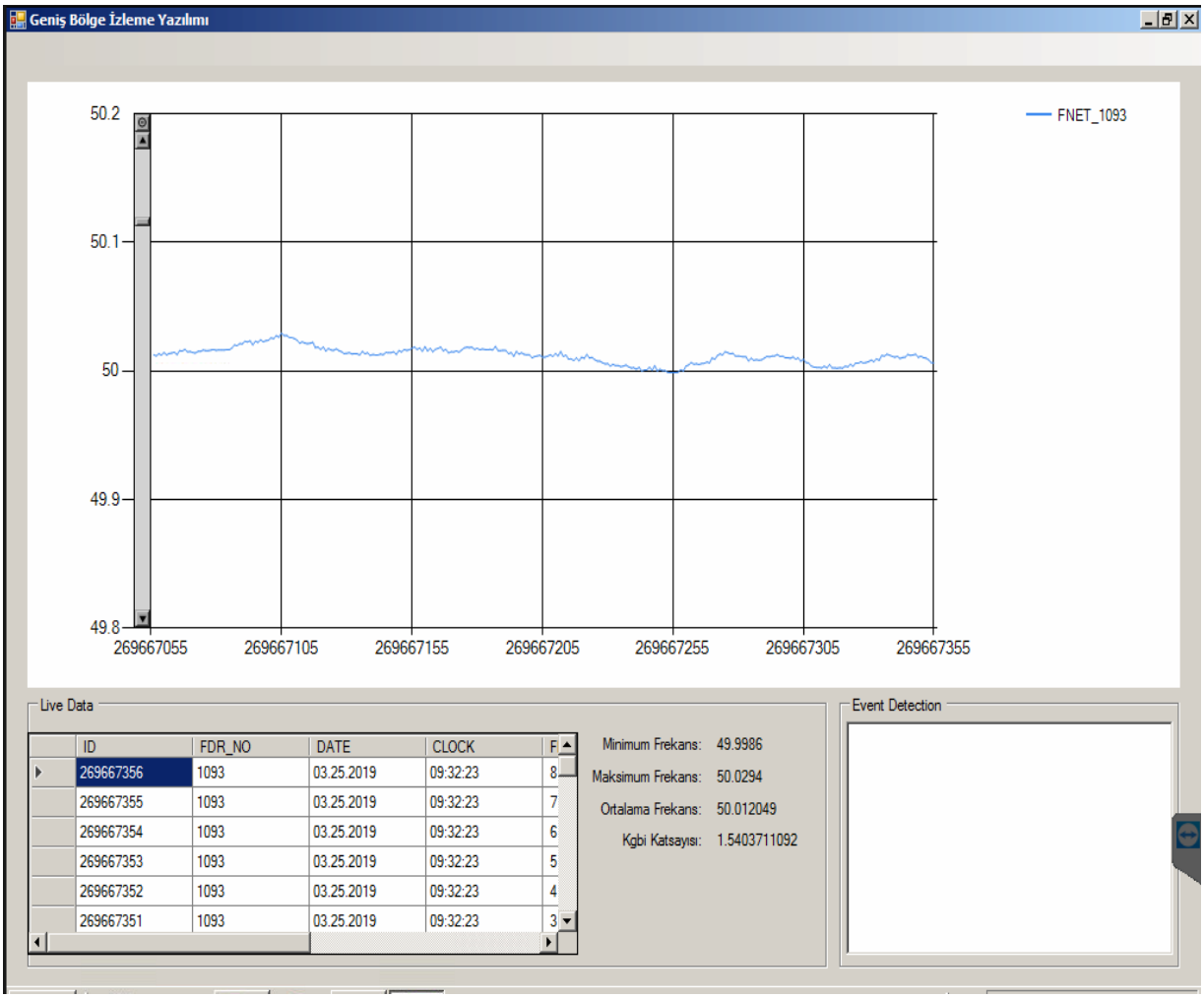
Algoritmanın tespit ettiği olaya ilişkin detaylar şu şekildedir;

Şekil 2.28. 01.01.2018 Tarihli Test Edilen Veri Sonucu

İndex	1702
Tespit Edilen Olay Anı	01-Jan-2018 10:47:20
Frekans	60.0193 Hz
Kgbi Referans Değeri	2.049316
Olayın Gerçekleşme Anı	01-Jan-2018 10:47:20
Nominal Frekans	60 Hz

2.9. Geniş Bölge İzleme Yazılımı

Verilerin filtrelenmesi, işlenmesi, depolanması ve yorumlanması gibi işlemlerin tamamı Visual Studio platformunda c#.net dilinde hazırlanan yazılım vasıtasıyla gerçekleştirilmektedir.



Şekil 2.29. Geniş Bölge İzleme Yazılımı

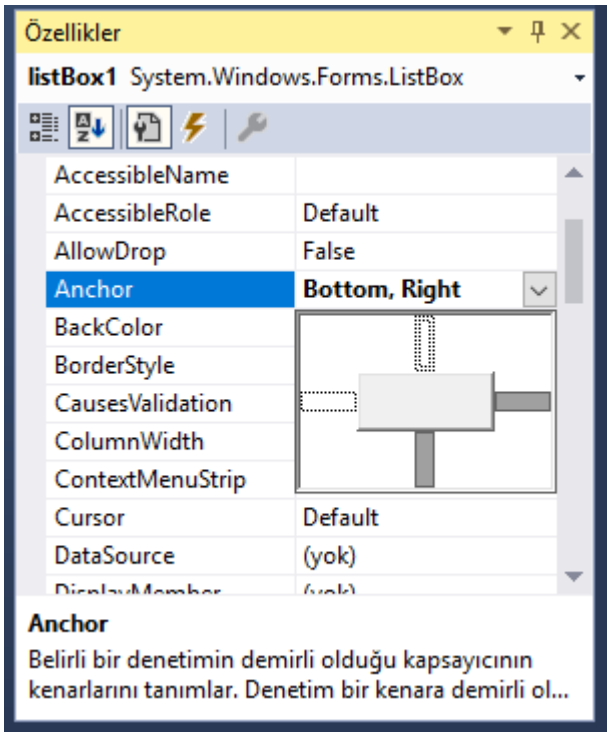
Geniş Bölge İzleme yazılımı tek form üzerinde, farklı işlemler gruplandırılarak tasarlanmıştır. Yazılım içerisinde her FDR cihazını için ayrı Thread oluşturularak verilerin güvenli bir şekilde ayrıştırılması ve eksiksiz bir şekilde birbirine karışmadan seçilmesi sağlanmıştır.

2.9.1. Yazılım Görsel Tasarımı ve Kodlaması

Form üzerinde grafik için bir adet chart, verilerin gösterildiği bir adet datagirdview, algoritmanın ihtiyaç duyduğu bilgilerin gösterilmesi çin 8 adet label ve tespit edilen olayların kayıt edilebilmesi için bir adet listbox kullanılmıştır.

Yazılımın tasarımında sadelik ve erişilebilirlik ön planda tutulmuştur. Algoritmadaki tüm akışlar ve veriler ile gerçekleşen olaylar tek pencereden izlenebilir olarak tek form şeklinde hazırlanmıştır. Form üzerinde frekans verisinin görsel olarak takibinin yapılabilmesi için Chart objesi koyulmuş ve ilgili FDR cihazına göre farklı renklerde çizgisel olarak veriler gösterilmiştir. Chart objesine ait koleksiyonlar her FDR cihazının kendine ait Thread bölümünde işlendiğinden bağımsız olarak hareket edebilmekte ve bu sayede herhangi bir FDR cihazından veri kaybı olması durumunda bile izlenilebilirliği devam etmektedir.

Yazılımın farklı çözünürlüklerde ve farklı boyutlarda da çalışabilmesi için form üzerindeki tüm nesnelerin Anchor özelliği alt ve sağ kısım dinamik, sol ve üst kısım sabit olacak şekilde düzenlenmiştir.



Şekil 2.30. Nesnelerin Anchor Özelliği

Tasarıma ait dizayn kodları şu şekildedir;

```
// menuStrip1
        //
        this.menuStrip1.Location = new System.Drawing.Point(0, 0);
        this.menuStrip1.Name = "menuStrip1";
        this.menuStrip1.Size = new System.Drawing.Size(815, 24);
        this.menuStrip1.TabIndex = 0;
        this.menuStrip1.Text = "menuStrip1";
        //
        // listBox1
        //
        this.listBox1.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom     |
System.Windows.Forms.AnchorStyles.Right)));
        this.listBox1.FormattingEnabled = true;
        this.listBox1.Location = new System.Drawing.Point(6, 19);
        this.listBox1.Name = "listBox1";
        this.listBox1.Size = new System.Drawing.Size(280, 173);
        this.listBox1.TabIndex = 2;
        //
        // groupBox1
        //
        this.groupBox1.Anchor                                                   =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom     |
System.Windows.Forms.AnchorStyles.Right)));
        this.groupBox1.Controls.Add(this.listBox1);
        this.groupBox1.Location = new System.Drawing.Point(505, 299);
        this.groupBox1.Name = "groupBox1";
```

```
            this.groupBox1.Size = new System.Drawing.Size(292, 202);
            this.groupBox1.TabIndex = 3;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Event Detection";
            //
            // groupBox2
            //
            this.groupBox2.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Bottom    |
System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.groupBox2.Controls.Add(this.label9);
            this.groupBox2.Controls.Add(this.label10);
            this.groupBox2.Controls.Add(this.label8);
            this.groupBox2.Controls.Add(this.dataGridView1);
            this.groupBox2.Controls.Add(this.label7);
            this.groupBox2.Controls.Add(this.label1);
            this.groupBox2.Controls.Add(this.label6);
            this.groupBox2.Controls.Add(this.label2);
            this.groupBox2.Controls.Add(this.label5);
            this.groupBox2.Controls.Add(this.label3);
            this.groupBox2.Controls.Add(this.label4);
            this.groupBox2.Location = new System.Drawing.Point(18, 299);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(477, 202);
            this.groupBox2.TabIndex = 4;
            this.groupBox2.TabStop = false;
```

```
        this.groupBox2.Text = "Live Data";
        //
        // label8
        //
        this.label8.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
        this.label8.AutoSize = true;
        this.label8.BackColor = System.Drawing.SystemColors.Control;
        this.label8.Location = new System.Drawing.Point(379, 100);
        this.label8.Name = "label8";
        this.label8.Size = new System.Drawing.Size(13, 13);
        this.label8.TabIndex = 13;
        this.label8.Text = "0";
        //
        // dataGridView1
        //
        this.dataGridView1.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
        this.dataGridView1.Location = new System.Drawing.Point(0, 27);
        this.dataGridView1.Name = "dataGridView1";
        this.dataGridView1.Size = new System.Drawing.Size(263, 165);
        this.dataGridView1.TabIndex = 0;
        //
        // label7
```

```
            //
            this.label7.Anchor                                                  =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top     |
System.Windows.Forms.AnchorStyles.Right)));
            this.label7.AutoSize = true;
            this.label7.BackColor = System.Drawing.SystemColors.Control;
            this.label7.Location = new System.Drawing.Point(379, 77);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(13, 13);
            this.label7.TabIndex = 12;
            this.label7.Text = "0";
            //
            // label1
            //
            this.label1.Anchor                                                  =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top     |
System.Windows.Forms.AnchorStyles.Right)));
            this.label1.AutoSize = true;
            this.label1.BackColor = System.Drawing.SystemColors.Control;
            this.label1.Location = new System.Drawing.Point(281, 29);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(92, 13);
            this.label1.TabIndex = 6;
            this.label1.Text = "Minimum Frekans:";
            //
            // label6
            //
```

```
        this.label6.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top          |
System.Windows.Forms.AnchorStyles.Right)));

        this.label6.AutoSize = true;

        this.label6.BackColor = System.Drawing.SystemColors.Control;

        this.label6.Location = new System.Drawing.Point(379, 53);

        this.label6.Name = "label6";

        this.label6.Size = new System.Drawing.Size(13, 13);

        this.label6.TabIndex = 11;

        this.label6.Text = "0";

        //

        // label2

        //

        this.label2.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top          |
System.Windows.Forms.AnchorStyles.Right)));

        this.label2.AutoSize = true;

        this.label2.BackColor = System.Drawing.SystemColors.Control;

        this.label2.Location = new System.Drawing.Point(272, 53);

        this.label2.Name = "label2";

        this.label2.Size = new System.Drawing.Size(101, 13);

        this.label2.TabIndex = 7;

        this.label2.Text = "Maksimum Frekans:";

        //

        // label5

        //

        this.label5.Anchor                                                    =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top          |
System.Windows.Forms.AnchorStyles.Right)));
```

```
            this.label5.AutoSize = true;
            this.label5.BackColor = System.Drawing.SystemColors.Control;
            this.label5.Location = new System.Drawing.Point(379, 29);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(13, 13);
            this.label5.TabIndex = 10;
            this.label5.Text = "0";
            //
            // label3
            //
            this.label3.Anchor                                          =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top          |
System.Windows.Forms.AnchorStyles.Right)));
            this.label3.AutoSize = true;
            this.label3.BackColor = System.Drawing.SystemColors.Control;
            this.label3.Location = new System.Drawing.Point(280, 77);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(93, 13);
            this.label3.TabIndex = 8;
            this.label3.Text = "Ortalama Frekans:";
            //
            // label4
            //
            this.label4.Anchor                                          =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top          |
System.Windows.Forms.AnchorStyles.Right)));
            this.label4.AutoSize = true;
            this.label4.BackColor = System.Drawing.SystemColors.Control;
```

```
this.label4.Location = new System.Drawing.Point(298, 100);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(75, 13);
this.label4.TabIndex = 9;
this.label4.Text = "Kgbi Katsayısı:";
//
// chart1
//
this.chart1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right)));
chartArea3.Name = "ChartArea1";
this.chart1.ChartAreas.Add(chartArea3);
legend3.Name = "Legend1";
this.chart1.Legends.Add(legend3);
this.chart1.Location = new System.Drawing.Point(18, 36);
this.chart1.Name = "chart1";
series3.ChartArea = "ChartArea1";
series3.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
series3.Legend = "Legend1";
series3.Name = "FNET_1092";
this.chart1.Series.Add(series3);
this.chart1.Size = new System.Drawing.Size(779, 257);
this.chart1.TabIndex = 5;
this.chart1.Text = "chart1";
```

```
            //
            // timer1
            //
            this.timer1.Enabled = true;
            this.timer1.Interval = 1;
            this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
            //
            // label9
            //
            this.label9.Anchor                                                  =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top         |
System.Windows.Forms.AnchorStyles.Right)));
            this.label9.AutoSize = true;
            this.label9.BackColor = System.Drawing.SystemColors.Control;
            this.label9.Location = new System.Drawing.Point(379, 123);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(13, 13);
            this.label9.TabIndex = 15;
            this.label9.Text = "0";
            //
            // label10
            //
            this.label10.Anchor                                                 =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top         |
System.Windows.Forms.AnchorStyles.Right)));
            this.label10.AutoSize = true;
            this.label10.BackColor = System.Drawing.SystemColors.Control;
            this.label10.Location = new System.Drawing.Point(311, 123);
```

```
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(62, 13);
this.label10.TabIndex = 14;
this.label10.Text = "Maks Eğim:";
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(815, 513);
this.Controls.Add(this.chart1);
this.Controls.Add(this.groupBox2);
this.Controls.Add(this.groupBox1);
this.Controls.Add(this.menuStrip1);
this.MainMenuStrip = this.menuStrip1;
this.Name = "Form1";
this.Text = "Geniş Bölge İzleme Yazılımı";
this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
this.Load += new System.EventHandler(this.Form1_Load);
this.groupBox1.ResumeLayout(false);
this.groupBox2.ResumeLayout(false);
this.groupBox2.PerformLayout();
((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();
```

2.9.2. Algoritma Kodları

Matlab ortamında test edilen algoritmalar, Windows ortamında çalışabilecek şekilde paket program haline getirilmesi için Visual Studio ortamında C# programlama dilinde programlanmıştır.

FDR cihazı tarafından işlenmek üzere yazılımın okuduğu veriler, algoritma sonrası işlenen veriler ve yakalanan olaylar Microsoft SQL veri tabanına kaydedilmiştir. Ayrıca yakalanan olaylara ilişkin detaylar, bilgi amaçlı belirtilen mail adreslerine gönderilecek şekilde kodlama yapılmıştır.

Yazılımda C#.net dilinde kullanılan kütüphaneleri gösteren kod şu şekildedir;

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Data.SqlClient;
using System.Windows.Forms.DataVisualization.Charting;
```

Algoritmanın çalışabilmesi için veri tabanına erişim esastır. Bu sebeple ilgili veri tabanına erişimin yazılımın çalışması esnasında kontrol edilmesi gerekmektedir. Veritabanında kontrol için ilgili kod yazılmıştır. İlgili kod fonksiyon haline getirilerek, ihtiyaç duyulduğu zamanlarda bu fonksiyon çağırılarak SQL veri tabanına erişim test edilmiştir ve böylelikle tekrar tekrar aynı kodun yazılması engellenmiştir.

SQL veritabanına bağlantının kontrol edildiği C#.net dilinde hızırlanmış fonksiyon kodları şu şekildedir;

```
private void SqlTestEt()
{

    SqlConnection baglanti;


baglanti = new SqlConnection("Data Source=.; Initial Catalog=FNET; Integrated
Security=true");

    try
    {
        baglanti.Open();
        MessageBox.Show("Veritabanına bağlantı başarılı.");
        baglanti.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("veritabanına bağlantı başarısız!");
        MessageBox.Show(ex.ToString());
        baglanti.Close();
    }

}
```

Veri tabanı içerisinde her FDR cihazı için ayrı tablo oluşturulmuştur. Bu tablonun silinmesi ya da başka bir yere kopyalanması sonucu veri kaydının hatasız bir şekilde devam edebilmesi için, programın açılışında veri tabanına bağlanılarak tabloların yeniden oluşturulmasına yönelik kod yazılmıştır. Veri tabanı içerisinde halihazırda ilgili tablonun bulunması durumlarında program hata vereceğinden try-catch kullanılarak hata oluşmadan programın akışı sağlanmıştır.

```
try
{
    using (SqlCommand cmd = new SqlCommand("CREATE TABLE [dbo].[FNET_1092]("
                + "[ID] [int] IDENTITY(1,1) NOT NULL,"
                + "[FDR_NO] [nvarchar](10) NULL,"
                + "[DATE] [date] NULL,"
                + "[CLOCK] [time](1) NULL,"
                + "[FRAME] [nvarchar](10) NULL,"
                + "[FREQUENCY] [nvarchar](10) NULL,"
                + "[VOLTAGE] [nvarchar](10) NULL,"
                + "[VOLTAGE_ANGLE] [nvarchar](10) NULL,"
                + "CONSTRAINT [FDR_1092] PRIMARY KEY CLUSTERED "
                + "("
                + "[ID] ASC"
                + ")WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]"
```

```
                + ") ON [PRIMARY]", baglanti))
    {
        cmd.Connection.Open();
        cmd.ExecuteNonQuery();
        cmd.Connection.Close();
    }
}
catch (Exception)
{

}
```

Yazılımın çalışması esnasında yapılan bu kontrollerin olumlu geri dönüşü sonrasında yazılım, FDR cihazlarından gönderilen bilgiler okumaya başlayacaktır. FDR cihazlarından gelen bilgilerin birbiri ile karışmaması için her cihaz farklı port üzerinden haberleştirilmiştir. Yazılım her FDR cihazı için ayrı Thread içerisinde port dinlemesi yaptığı için aynı anda birden çok port dinlenebilmekte ve bu sayede veri kaybı oluşmamaktadır.

```
TcpListener dinleyiciSoket = new TcpListener(System.Net.IPAddress.Any, 9592);
dinleyiciSoket.Start();
Socket istemciSoketi = dinleyiciSoket.AcceptSocket();
NetworkStream agAkisi = new NetworkStream(istemciSoketi);
BinaryReader binaryOkuyucu = new BinaryReader(agAkisi);
```

FDR cihazından gelen veri 54 karakterden oluşmaktadır. Verinin doğru bir şekilde elde edilebilmesi için boyut kontrolü yapılmıştır. Veri 54 karaktere ulaştıktan sonra Substring fonksiyonuyla anlamlı parçalara ayrıldıktan sonra veri tabanındaki ilgili yerlere kayıt edilmektedir. Kayıt işlemi olumlu bir şekilde sonlandıktan sonra kontrol değişkenleri başlangıç değerlerine getirilerek döngü sağlanmaktadır.

```
if (bufferStart == true)
{
    gelen_veri = binaryOkuyucu.ReadChar();
    if (((int)gelen_veri) == 1)
    {
        if (bufferVeri.Length == 54)
        {

            //kayıt
            try
            {
```

```
                SqlCommand cmd = new SqlCommand("INSERT INTO dbo.FNET_1092 (FDR_NO,
DATE, CLOCK, FRAME, FREQUENCY, VOLTAGE, VOLTAGE_ANGLE) VALUES (@FDR_NO, @DATE, @CLOCK,
@FRAME, @FREQUENCY, @VOLTAGE, @VOLTAGE_ANGLE)", baglanti);
                cmd.Parameters.AddWithValue("@FDR_NO", bufferVeri.Substring(0, 4));
                cmd.Parameters.AddWithValue("@DATE", bufferVeri.Substring(5, 2) + "." +
bufferVeri.Substring(7, 2) + "." + bufferVeri.Substring(9, 2));
                cmd.Parameters.AddWithValue("@CLOCK", bufferVeri.Substring(12, 2) + ":"
+ bufferVeri.Substring(14, 2) + ":" + bufferVeri.Substring(16, 2));
                cmd.Parameters.AddWithValue("@FRAME", bufferVeri.Substring(19, 2));
                cmd.Parameters.AddWithValue("@FREQUENCY", bufferVeri.Substring(22, 7));
                cmd.Parameters.AddWithValue("@VOLTAGE", bufferVeri.Substring(38, 8));
                cmd.Parameters.AddWithValue("@VOLTAGE_ANGLE", bufferVeri.Substring(47,
6));

                if (baglanti.State == ConnectionState.Closed)
                {
                    baglanti.Open();
                }
                cmd.ExecuteNonQuery();
                baglanti.Close();
            }
            catch (SqlException)
            {

            }

            bufferVeri = "";
        }
        bufferVeri = "";
    }
    else
    {
        bufferVeri += gelen_veri;
    }
}
else
{
    gelen_veri = binaryOkuyucu.ReadChar();
    if (((int)gelen_veri) == 1)
    {
        bufferStart = true;
    }
}
```

Yukarıdaki işlemler sonrasında program, ilgili FDR cihazından gelen bilgileri, tanımlanan portu dinleyerek elde etmiş ve verinin doğru bir şekilde elde edildiğini kontrol ettikten sonra veri tabanındaki ilgili yere kaydetmiştir. Kaydedilen bu değerler algoritma öncesi elde edilmiş ham verilerdir.

Olay yakalama algoritması son 30sn içerisindeki veri üzerinden analiz yaptığından ve her saniyede 10 örnekleme elde edildiğinden dolayı veri tabanındaki son 300 veri üzerinden işlem yapılmaktadır. İlgili kod dizimi fonksiyon haline getirilmiştir. Veri tabanından alınan 300 veri aynı zamanda form üzerindeki grafikte doğrusal olarak gösterilmektedir.

```
private void kayitGetir()
{
    SqlConnection baglanti;
    baglanti = new SqlConnection("Data Source=.; Initial Catalog=FNET; Integrated
Security=true");
    baglanti.Open();
    SqlDataAdapter verial = new SqlDataAdapter("SELECT TOP 300
[ID],[FDR_NO],[DATE],[CLOCK],[FRAME],[FREQUENCY],[VOLTAGE],[VOLTAGE_ANGLE] FROM
[FNET].[dbo].[FNET_1092] Order By ID DESC", baglanti);

    DataTable dset = new DataTable();
    verial.Fill(dset);

    chart1.DataSource = dset;
    chart1.Series["FNET_1092"].YValueMembers = "FREQUENCY";
    chart1.Series["FNET_1092"].XValueMember = "ID";
    chart1.DataBind();
}
```

Ham frekans verisinin algoritmada kullanılacak olan son 30 sn içerisindeki 300 veri çekilerek DataTable türünde değişkene tanımlanmıştır. Veriler çekilirken veritabanından SQL sorgusu ile çekilmiştir. Son 30 sn verisi için SELECT TOP 300 sorgusu ile sıralamanın sonundan veri çekilmiş, Order By ID DESC sorgusu ile sıralamanın sondan geriye doğru olması sağlanmıştır.

Verilerin daha anlamlı hale getirilmesi için Matlab ortamında test edilen filtrelemeler ve algoritma, aynı şekilde c# ortamında da hazırlanmıştır. Ham veri üzerinde ortalama filtre yöntemi, eğim belirleme ve kgbi katsayısı belirleme algoritmaları uygulanmıştır.

```
decimal[] ortalamaFiltre = new decimal[300];

ortalamaFiltre[0] = Convert.ToDecimal(dset.Rows[0]["FREQUENCY"]);
ortalamaFiltre[1] = Convert.ToDecimal(dset.Rows[1]["FREQUENCY"]);
ortalamaFiltre[2] = Convert.ToDecimal(dset.Rows[2]["FREQUENCY"]);
ortalamaFiltre[3] = Convert.ToDecimal(dset.Rows[3]["FREQUENCY"]);
ortalamaFiltre[4] = Convert.ToDecimal(dset.Rows[4]["FREQUENCY"]);
ortalamaFiltre[5] = Convert.ToDecimal(dset.Rows[5]["FREQUENCY"]);
```

```
ortalamaFiltre[6] = Convert.ToDecimal(dset.Rows[6]["FREQUENCY"]);
ortalamaFiltre[7] = Convert.ToDecimal(dset.Rows[7]["FREQUENCY"]);
ortalamaFiltre[8] = Convert.ToDecimal(dset.Rows[8]["FREQUENCY"]);



for (int i=9 ; i< dset.Rows.Count-1 ; i++)
{
    ortalamaFiltre[i] = (Convert.ToDecimal(dset.Rows[i]["FREQUENCY"])+
Convert.ToDecimal(dset.Rows[i-1]["FREQUENCY"])
        + Convert.ToDecimal(dset.Rows[i-2]["FREQUENCY"])+
Convert.ToDecimal(dset.Rows[i-3]["FREQUENCY"])
        + Convert.ToDecimal(dset.Rows[i-4]["FREQUENCY"])+
Convert.ToDecimal(dset.Rows[i-5]["FREQUENCY"])
        + Convert.ToDecimal(dset.Rows[i-6]["FREQUENCY"])+
Convert.ToDecimal(dset.Rows[i-7]["FREQUENCY"])
        + Convert.ToDecimal(dset.Rows[i-8]["FREQUENCY"])+
Convert.ToDecimal(dset.Rows[i-9]["FREQUENCY"]))/10;

}
```

Ortalama filtre yöntemi ile frekans üzerindeki dalgalanmalar azaltılarak filtreleme yapılmıştır. Öncelikle filtrelenen verinin kaydedileceği ortalamaFiltre adında decimal türünde 300 boyutluk dizi değişkeni oluşturulmuştur. 10'lu ortalama alındığı için ilk 9 verinin filtrelenmesi yapılmamıştır. 10 adet veri elde edildikten sonra filtreleme işlemi başlatılmıştır.

For döngüsü ile datatable boyutunda döngü sağlanmıştır. Döngü içerisinde filtreleme işlemi yapıldıktan sonra veri türü değiştirilerek dizi olarak veriler kaydedilmiştir.

Frekans verisinin Ortalama Filtre yöntemiyle filtrelendikten sonra Kgbi referans katsayısına yansıtılmak üzere eğimi alınıp belirtilen değere yakınsatılacaktır.

```
decimal[] egimFiltre = new decimal[300];
 egimFiltre[0] = 0;
 for(int i=1; i < ortalamaFiltre.Length - 1; i++)
 {
     egimFiltre[i] = Math.Abs(ortalamaFiltre[i] - ortalamaFiltre[i-1]);
 }
```

Pozitif eğim verisi ile frekansın anlık olarak değişimi çok daha okunabilir bir hale getirilmiştir. Frekansın tepkisinin yönünden ziyade büyüklüğü dikkate alındığı için math

kütüphanesi içerisinde bulunan abs fonksiyonu ile eğimleri mutlak değeri alınarak diziye kaydedilmiştir.

Eğim çıkarma algoritmasıyla tepki büyüklüğü hesaplanan frekans verileri ile kgbi katsayısı çarpılmadan önce eğim dizisinin 2 değerine normalizasyonu yapılarak Kgbi üzerindeki etkisinin maksimum iki katına çıkarma olarak sınırlandırılması sağlanmıştır.

```
decimal[] normalizasyon = new decimal[300];
normalizasyon [0] = 0;
decimal maksEgim = egimFiltre.Max();

for (int i = 1; i < egimFiltre.Length - 1; i++)
{
    normalizasyon [i] = egimFiltre[i] / (maksEgim / 2);
}
```

Kgbi referans katsayısı, olay tespitinde kullanılan değişendir. Bu katsayı, 30sn'lik yani 300 adet verilik belirlenmiş hareket eden çerçeve içerisindeki frekansın ortalamasının o çerçevenin dağılımına göre oranını hesaplayan, bu şekilde çerçeve içerisindeki tepkilerin nominal frekans değerine göre değil de ortalama frekans verisine göre büyüklüğü ölçen bir katsayı olarak belirlenmiştir. Ayrıca frekansın ani tepkilerinin katsayı üzerinde etki oluşturması adına yakınsamış eğim dizisi ile çarpılmaktadır.

```
decimal minf = Convert.ToDecimal(dset.AsEnumerable().Min(row => row["FREQUENCY"]));
decimal maxf = Convert.ToDecimal(dset.AsEnumerable().Max(row => row["FREQUENCY"]));
decimal ortf = ortalamaFiltre.Average();
decimal kgbi = (maxf - minf) / ortf;
kgbi = kgbi * normalizasyon[normalizasyon.Length];
```

Kgbi verisi, olayın türünden bağımsız olarak, olayın şiddetine göre büyüyen bir sayısal değere sahiptir. Olay analizi yapılırken enterkonnekte şebekede bulunan anlık güç

miktarı bilinmediğinden kgbi referans değeri belirlenirken, geçmişte yaşanmış olaylardaki frekans bilgileri analiz edilerek sınır değer belirlenmiştir.

Çerçeve içerisindeki Kgbi referans katsayısının, çerçevenin ortalama değerinin 10 katına çıktığı anda olay tespit edilmektedir. Olayın tespit edilmesinden sonra frekans bilgisinden faydalanılarak olayın türü analiz edilmektedir. Çerçeve içerisindeki maksimum frekans değeri olayın öncesinde ise olay türü Generatör Devreden Çıkması olarak belirlenmektedir. Yük Devreden Çıkması olayında ise bu durumun tersi gerçekleşmektedir. Olay yakalama algoritmasına ait C#.net kodları şu şekildedir;

```
for (int i = 0; i < kgbi.Length; i++)
{

    if (kgbi[i] > kgbi.Average()*10)
    {
        listBox1.Items.Add("Olay tespit edildi: " + DateTime.UtcNow );
    }

}
```

Olay türü belirleme algoritması ait C#.net kodları şu şekildedir;

```
for (int i = 0; i < kgbi.Length; i++)
{

    decimal maks = kgbi.Max();

    if ( Array.IndexOf(kgbi,maks) > Array.IndexOf(kgbi, olayIndex))
    {
        olayTuru = "Generator Devreden Çıkması";
    }
    else
    {
        olayTuru = "Yük Devreden Çıkması";
    }
}
```

3. BULGULAR

Hazırlanan yazılım vasıtasıyla FDR cihazlarından gelen verilerin algoritmada oluşturulan kontrol sayesinde hatasız bir şekilde kaydedildiği görülmüştür. Olası internet kesintisi sonrasında bile bir sonraki veri üzerinden kayıt devam etmekte, iki veri bilgisi birbirine karışmamaktadır.

Geçmişte meydana gelen olaylar üzerinde algoritma test edildiğinde %100 e yakın sonuçlar elde edilmiştir. Algoritmanın tespit ettiği olay anı ile olayın gerçekleşme anı arasında en fazla 1 saniye fark oluşmaktadır.

Algoritma sonrası oluşturulan Kgbi referans katsayısının grafiğine bakıldığında, olayın meydana geldiği zaman çevresinde ciddi anlamda farklılık oluşturduğu, bu durumun da olayları tespit etme aşamasında başarı oranının yüksek olduğunu göstermektedir. Algoritmanın test edildiği örnek sayısının az olmasına karşın, Kgbi değerinin, stabil frekans zamanından ciddi anlamda farklılaşması farklı örneklerdeki başarı oranın yüksek olacağını göstermektedir.

Olay tespitinde kullanılan Kgbi referans değerinin örnek verilerde ki olay anı değerlerinin, çerçevenin ortalama değerinin yaklaşık 15-20 katına çıktığı gözlemlenmiştir. Şebekede meydana gelen tüm olayların tespit edilebilmesi adına ve normal şartlarda Kgbi referans değeri ortalamanın yaklaşık 1-2 katı şeklinde ilerlediğinden güvenli bölge olarak ortalama değerin 10 katı referans alınmıştır.

4. SONUÇLAR VE TARTIŞMA

Bu çalışmada, enterkonnekte şebekelerde meydana gelen ve frekansta bozulmalara sebep olan olaylar, şebeke frekansı ile tespit edilmeye çalışılmıştır. Bu hususta frekans verisi üzerinde filtreleme ve matematiksel işlemler yapılarak olay tespitine yönelik daha anlamlı ve okunabilir hale getirilmeye çalışılmıştır.

Oluşturulan algoritmanın modelleme ve test işlemleri MATLAB ortamında gerçekleştirilip olup sonrasında Visual Studio ortamında C# dilinde Windows ortamında çalışan yazılım haline getirilmiştir.

Algoritmaya tabi tutulan frekans ve zaman bilgileri Microsoft SQL Studio ortamında MSSQL Veritabanında saklanmıştır. Algoritmanın tespit ettiği olaylar yine aynı Veritabanında farklı bir tablo üzerinde, sonradan inceleme yapılabilmesi adına kaydedilmiştir.

Bu tez çalışması sırasında oluşturulan algoritmanın temeli 1 adet ulusal bildiride yayınlanmıştır. Bu bildiri 216 yılında Türkiye'nin Tokat ilçesinde bulunan Gaziosmanpaşa üniversitesinde "Elektrik-Elektronik ve Bilgisayar (EEB2016)" konulu sempozyumda sunulmuştur.

Tablo 4.1. Örnek Veri Sonuçları

No	Olayın Türü	Olay Gerçekleşme Zamanı (UTC)	Tespit Edilen Olay Anı (UTC)	Sapma (sn)
1	Yük Atması	25.02.2012 23:09:00	25.02.2012 23:09:00	0
2	Yük Atması	01.01.218 10:47:20	01.01.2018 10:47:20	0
3	Generatör Atması	03.27.2019 13:20:54	03.27.2019 13:20:54	0
4	Generatör Atması	03.13.2019 23:50:54	03.13.2019 23:50:55	1
5	Yük Atması	01.03.2017 11:01:25	01.03.2017 11:01:25	0

5. ÖNERİLER

Yapılan araştırmalar sonucunda elde edilen hesaplamalar, algoritmalar ve tespitler sonucunda enerjinin üretiminden tüketimine kadar bütün aşamalarında meydana gelen aksaklıkların tespiti için yeni bir bakış açısı getirilmiştir. Algoritmadaki katsayılar değiştirilerek ve geliştirilerek farklı yöntemlerle olay tespiti yapılabilir.

Algoritma, Microsoft Visual Studio ile C# programlama dilinde hazırlanan yazılıma entegre edilmiştir. Farklı derleyiciler ve yazılım dilleriyle geliştirilip farklı programlara entegre edilebilir.

Enterkonnekte şebekede farklı noktalara yerleştirilen FDR cihazlarıyla veri tabanı zenginleştirilebilir. Bu sayede olay yakalama olasılığı arttırılarak daha fazla test ortamı oluşturulabilir.

Algoritma geliştirilerek olay büyüklüğü ve olayın meydana geldiği yer konusunda çalışmalar yapılabilir. Tespit edilen olaylar sonrası alınabilecek önlemler konusunda analiz yapan çalışmalar geliştirilebilir.

Tez içeriğindeki benzer yaklaşım ile bir başka şebeke bozulması olan osilasyon tespitine yönelik çalışmalar yapılabilir.

Mevcut çalışmanın devamında enterkonnekte şebekede meydana gelen ve tespit edilen olaylar ile, frekansın olaylara karşı tepkisi analiz edilerek sisteme öğretilebilir hale getirilebilir. Bu sayede frekans değişimi üzerinden olaylar frekans tahmini ile önceden tespit edilebilir ve önlenebilir hale getirilebilir.

6. KAYNAKLAR

1. Mesleki ve Teknik Açık Öğretim Okulu, Enerji Üretimi, İletimi ve Dağıtımı, 1991, Ankara.

2. Xiangping, M., Qingxian, G., Lei, F., Wen, Z. ve Weijun, Z., PI Fuzzy Sliding Mode Load Frequency Control of Multiarea Interconnected Power Systems, IEEE International Symposium on Intelligent Control, Ekim 2003, Houston USA.

3. Darçın, O., Güç Sistemlerinde Yük Frekans Kontrolü ve Sinir Ağlı Kontrol Uygulaması, Yüksek Lisans Tezi, K.K.Ü., Fen Bilimleri Enstitüsü, Kırıkkale, 2004.

4. Tuttokmağı, Ö. Ve Kaygusuz, A., Büyük Ölçekli Elektrik Kesintilerinin İncelenmesi, BEÜ Fen Bilimleri Dergisi, 8, 2 (2019) 664-676.

5. Chunchun, X., Zhian, Z., Virgilio, C., Richard, C. ve Yilu, L., Practical Issues in Frequency Disturbance Recorder Design for Wide-Area Monitoring, Electrical Power Quality and Utilisation, 11,1 (2005) 69-75.

6. Yingchen, Z., Penn, M., Tao, X., Lang, C., Yanzhu, Y., Zhongyu, W., Zhiyong, Y., Lei, W., Jason, B., Jon, B., Richard, B. ve Yilu, L., Wide-Area Frequency Monitoring Network (FNET) Architecture and Applications, IEEE Transactions on Smart Grid, 1,2 (2010) 159-167.

7. Zhiyong, Y., Tao, X., Yingchen, Z., Lang, C., Penn, N., Matthew, G. ve Yilu, L., Inter-area oscillation analysis using wide area voltage angle measurements from FNET, IEEE PES General Meeting, Haziran 2010, Providence, RI, USA.

8. Jiahui, G.,Ye, Z., Yilu, L., Marcus, Y., Philip, I., Aleksandar, D. ve Patrick, W., Design and implementation of real-time off-grid detection tool based on FNET/GridEye, IEEE PES General Meeting | Conference & Exposition, Haziran 2014, National Harbor, MD, USA.

9. Sterpu, S., Rossignol, S. ve Prestat, B., Detection and Location of Generation Trips in Large Transmission Grids, IEEE Power Engineering Society General Meeting, Haziran 2007, Tampa, FL, USA.

10. Do-In, K., Tae Yoon, C., Sung-Hwa, Y., Gyul, L. ve Yong-June, S., Wavelet-Based Event Detection Method Using PMU Data, IEEE Transactions on Smart Grid, 8,3 (2017) 1154-1162.

11. Seethalekshmi, K., Singh, N. ve Srivastava, C., A Synchrophasor Assisted Frequency and Voltage Stability Based Load Shedding Scheme for Self-Healing of Power System, IEEE Transactions on Smart Grid, 2,2 (2011) 221-230.

12. Le, X., Yang, C. ve Kumar, R., Dimensionality Reduction of Synchrophasor Data for Early Event Detection: Linearized Analysis, IEEE Transactions on Power Systems, 29,6 (2014) 2784-2794.

13. Lei, W., Jon, B., Jian, Z., Chun, X., Bruce, B., Richard, C. ve Yilu, L., Frequency Disturbance Recorder Design and Developments, IEEE Power Engineering Society General Meeting, Haziran 2007, Tampa, FL, USA.

14. Teker, Z., Doğrudan C#.NET Fundamentals Desktop Database Web, Godoro Yayınılık, İstanbul, 2017.

ÖZGEÇMİŞ

Mehmet Emre TERZİ, 1991 yılında Trabzon'un Of ilçesinde doğdu. İlköğretim ve ortaöğretimini Cumapazarı Beldesi'nde tamamlamış olup, 2013 yılında Karadeniz Teknik Üniversitesi Elektrik-Elektronik Mühendisliği Bölümünden mezun oldu. Aynı sene Karadeniz Teknik Üniversitesi Elektrik-Elektronik Mühendisliği bölümünde yüksek lisans eğitimine başladı. İyi derecede ingilizce bilen TERZİ, enterkonnekte şebekeler üzerine çalışmalarını sürdürmektedir.