T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FARKSAL GELİŞİM ALGORİTMASI KULLANILARAK ADAPTİF GÜRÜLTÜ GİDERİCİ

Tezi Hazırlayan
Nalân YİĞİT

Tezi Yöneten
Yrd. Doç. Dr. Nurhan KARABOĞA

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Haziran 2007
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FARKSAL GELİŞİM ALGORİTMASI KULLANILARAK ADAPTİF GÜRÜLTÜ GİDERİCİ

Tezi Hazırlayan
Nalân YİĞİT

Tezi Yöneten
Yrd. Doç. Dr. Nurhan KARABOĞA

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Haziran 2007
KAYSERİ

Yrd. Doç. Dr. Nurhan KARABOĞA danışmanlığında **Nalân YİĞİT** tarafından hazırlanan **"Farksal Gelişim Algoritması Kullanılarak Adaptif Gürültü Giderici"** adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalında **Yüksek Lisans** tezi olarak kabul edilmiştir.

15/06/2007

JÜRİ:

Başkan: Doç. Dr. M. Emin YÜKSEL

Üye :Doç. Dr. Recai KILIÇ

Üye :Yrd. Doç. Dr. Nurhan KARABOĞA

ONAY:

Bu tezin kabulü, Enstitü Yönetim Kurulunun 19/06/2007 tarih ve 2007/17-03 sayılı kararı ile onaylanmıştır.

20/06/2007

N. Ayyıldız

Prof. Dr. Nusret AYYILDIZ

Enstitü Müdürü

TEŞEKKÜR

Tez çalışmasının danışmanlığını yürüten ve yüksek lisans çalışmalarım boyunca bu çalışmanın gerçekleştirilmesinde anlayış, bilgi ve tecrübelerini esirgemeyen saygı değer hocam Yrd. Doç. Dr. Nurhan KARABOĞA'ya, ve tezi hazırlamamda beni yalnız bırakmayan sevgili arkadaşım Canan Aslıhan KOYUNCU'ya ve öğrenim hayatım boyunca maddi ve manevi desteğini esirgemeyen her zaman yanımda olan sevgili aileme teşekkürlerimi sunarım.

FARKSAL GELİŞİM ALGORİTMASI KULLANILARAK ADAPTİF GÜRÜLTÜ GİDERİCİ

Nalân YİĞİT
Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Haziran 2007
Tez Danışmanı: Yrd. Doç. Dr. Nurhan KARABOĞA

ÖZET

Bir adaptif süzgeç, bir maliyet fonksiyonunu optimize etmek için veya önceden belirlenen optimizasyon kriterlerini karşılamak için katsayıları ayarlanan ve zamanla değişen bir süzgeçtir. Adaptif süzgeçler, kendi parametrelerini otomatik olarak ayarlama kabiliyetine sahiptir ve tasarımları işaretin ya da gürültünün özelliklerinin önceki bilgilerine çok az ihtiyaç duyularak veya hiç gerek duyulmaksızın yapılmaktadır. Gürültü giderme birçok uygulamada çok fazla avantaja sahip olan optimal süzmenin bir çeşididir. İşaretin algılanamadığı ya da zayıf olduğu gürültü alanındaki noktalara yerleştirilen bir veya daha fazla sensörden elde edilmiş yardımcı ya da referans giriş kullanılır. Giriş süzülür ve işaret ve gürültünün her ikisini de içeren ana (primary) girişten çıkartılır. Sonuç olarak, gürültü zayıflatılır ve giderici tarafından elimine edilir.

Farksal Gelişim Algoritması basit, ama güçlü popülasyon tabanlı bir algoritmadır. Özellikle tamamen düzenlenmiş uzayda tanımlı ve gerçek değerli tasarım parametrelerini içeren fonksiyonları küresel olarak optimize etmek amacıyla kullanılan bir direkt araştırma algoritmasıdır. Bu çalışmada, Farksal Gelişim algoritması kullanılarak adaptif FIR gürültü giderici tasarlanmış ve bu algoritmanın performansı, üç farklı bilgi işareti için ve beş farklı süzgeç derecesi için, daha önce literatürde kullanılan klasik bir algoritma olan LMS (Least Mean Square-En Küçük Ortalama Kareler) algoritması ve bu algoritmadan türetilen NLMS (Normalized Least Mean Square-Normalize Edilmiş En Küçük Ortalama Kareler) algoritması kullanılarak araştırılmıştır. Adaptif FIR gürültü gidericinin farklı SNR (Signal to Noise- İşaret Gürültü Oranı) değerleri için başarımı ve tasarımda kullanılan Farksal Gelişim algoritmasının kontrol parametrelerinin değişiminin gürültü gidericiye etkisi incelenmiştir.

Anahtar Kelimeler: Farksal Gelişim Algoritması, Adaptif Süzgeç, Gürültü Giderme.

ADAPTIVE NOISE CANCELLER BY USING DIFFERENTIAL EVOLUTION ALGORITHMS

Nalân YİĞİT
Erciyes University, Graduate School of Natural and Applied Sciences
M.Sc. Thesis, June 2007
Thesis Supervisor: Assist. Prof. Dr. Nurhan KARABOĞA

ABSTRACT

An adaptive filter is a time-variant filter whose coefficients are adjusted in a way to optimize a cost function or to satisfy some predetermined optimization criterion. Adaptive filters have the ability of adjusting their own parameters automatically, and their design requires no prior knowledge of signal or noise characteristics. Noise cancelling is a variation of optimal filtering that is highly advantageous in many applications. It uses an auxiliary or reference input derived from one or more sensors located at points in the noise field where the signal is weak or undetectable. This input is filtered and subtracted from a primary input containing both signal and noise. As a result, the primary noise is attenuated or eliminated by cancellation.

Differential evolution algorithm is a simple, robust and population based algorithm. Especially, it is a direct search algorithm used to optimize functions as spherical, which involve design parameters described in completely regulated space and with real value. In this study, adaptive FIR (Finite Impulse Response) noise cancellation is designed by using differential evolution algorithm and performance of this algorithm is researched with three different information signals and five different filter degrees, also, a performance comparison is made with the LMS algorithm which is a classical algorithm used in literature and NLMS algorithm derived from LMS algorithm and the performance of the adaptive FIR noise cancellers for different SNR values parameter and the effect of the parameter values of Differential evolution algorithm used in noise cancellation is examined.

Keywords: Differential Evolution Algorithm, Adaptive Filter, Noise Cancellation.

İÇİNDEKİLER

KABUL VE ONAY i
TEŞEKKÜR.....ii
ÖZET…….……..……..…….……....iii
ABSTRACT……...……..……..……..…….……iv
KISALTMALAR……..……...……..……..…….……....viii
TABLOLAR LİSTESİ……..……..…….……...ix
ŞEKİLLER LİSTESİ……...……..……..….xi

1. BÖLÜM 1
GİRİŞ 1

2. BÖLÜM 5
ADAPTİF SÜZGEÇLER.....5
2.1.Adaptif Süzgeçler.....5
2.1.1.Süzgeç Yapısı.....8
2.1.2.Adaptif Süzgeçlerin Karakteristikleri 10
2.1.3. Adaptif Algoritma…….……11
2.1.4.Adaptif Süzgeçleme Uygulamaları……..……..12
2.2.Adaptif Gürültü Giderme 15
2.2.1.Adaptif Gürültü Giderici 16
2.3.LMS Algoritması 20
2.3.1.LMS Algoritmasının Kararlılığı.....20
2.3.2.LMS Algoritmasının Yapısı.....21
2.3.3.LMS Algoritmasının Öğrenme Eğrisi.....24
2.3.4.LMS Algoritmasının Dezavantajları 24
2.4.Normalize Edilmiş LMS Algoritması (NLMS) 24
2.5.SNR (Signal-to-Noise Ratio – İşaret Gürültü Oranı).....26

3. BÖLÜM 27
FARKSAL GELİŞİM ALGORİTMASI.....27
3.1.Evrimsel Algoritmalar.....27
3.2.Farksal Gelişim Algoritması 29

3.3.Algoritmanın Temel Adımları....31
3.3.1.Kodlama....34
3.3.2.Popülasyon Yapısı ve Parametre Sınırları....35
3.3.3.Mutasyon....36
3.3.4.Seçme....36
3.3.5.Yeniden Birleştirme....37
3.4.Farklı FG Algoritmaları....37
3.4.1.DE/rand/1/bin....38
3.4.2.DE/best/2/bin....39

4. BÖLÜM....41
ADAPTİF FIR SÜZGEÇLERDE GÜRÜLTÜ GİDERME....41
4.1.Adaptif Süzgeçler....41
4.2.FG ALGORİTMASI KULLANILARAK ADAPTİF GÜRÜLTÜ GİDERİCİ TASARIMI....42
4.2.1.Tasarım Hakkında Genel Bilgi....42
4.2.2.$\sin(2\pi t^2/8N^2)$ Bilgi İşareti Kullanılarak AGG Tasarımı....44
4.2.2.1.LMS Algoritması İle AGG Tasarımı:....45
4.2.2.2.NLMS Algoritması İle AGG Tasarımı:....46
4.2.2.3.FG Algoritması İle AGG Tasarımı:....47
4.2.3.$\sin(2\pi t^2/8N^2)+ \sin(2\pi t^2/4N^2)$ Bilgi İşareti Kullanılarak AGG Tasarımı:....47
4.2.3.1.LMS Algoritması İle AGG Tasarımı:....49
4.2.3.2.NLMS Algoritması İle AGG Tasarımı:....50
4.2.3.3.FG Algoritması İle AGG Tasarımı:....50
4.2.4.$\sin(2\pi t^2/8N^2)+\sin(2\pi t^2/4N^2)+\sin(2\pi t^2/2N^2)$ Bilgi İşareti Kullanılarak AGG Tasarımı:....51
4.2.4.1.LMS Algoritması İle AGG Tasarımı:....52
4.2.4.2.NLMS Algoritması İle AGG Tasarımı:....53
4.2.4.3.FG Algoritması İle AGG Tasarımı:....54
4.3.FARKLI SÜZGEÇ DERECELERİ İÇİN AGG TASARIMI....54
4.3.1.$\sin(2\pi t^2/8N^2)$ Bilgi İşareti Kullanılarak Farklı Süzgeç Dereceleri İle AGG Tasarımı:....54
4.3.1.1.İkinci Dereceden Süzgeç Kullanılarak AGG:....55

4.3.1.2.Dördüncü Dereceden Süzgeç Kullanılarak AGG: 56
4.3.1.3.Altıncı Dereceden Süzgeç Kullanılarak AGG: 58
4.3.1.4.Sekizinci Dereceden Süzgeç Kullanılarak AGG: 59
4.3.1.5.Onuncu Dereceden Süzgeç Kullanılarak AGG:..... 61
4.3.2.$\sin(2\pi t^2/8N^2)+\sin(2\pi t^2/4N^2)$ Bilgi İşareti Kullanılarak Farklı Süzgeç Dereceleri İle AGG Tasarımı:..... 63
4.3.2.1.İkinci Dereceden Süzgeç Kullanılarak AGG:..... 64
4.3.2.2.Dördüncü Dereceden Süzgeç Kullanılarak AGG: 65
4.3.2.3.Altıncı Dereceden Süzgeç Kullanılarak AGG: 66
4.3.2.4.Sekizinci Dereceden Süzgeç Kullanılarak AGG: 68
4.3.2.5.Onuncu Dereceden Süzgeç Kullanılarak AGG:..... 70
4.3.3.$\sin(2\pi t^2/8N^2)+\sin(2\pi t^2/4N^2)+\sin(2\pi t^2/2N^2)$ Bilgi İşareti Kullanılarak Farklı Süzgeç Dereceleri İle AGG Tasarımı:..... 72
4.3.3.1.İkinci Dereceden Süzgeç Kullanılarak AGG:..... 72
4.3.3.2.Dördüncü Dereceden Süzgeç Kullanılarak AGG: 74
4.3.3.3.Altıncı Dereceden Süzgeç Kullanılarak AGG: 75
4.3.3.4.Sekiz Dereceden Süzgeç Kullanılarak AGG: 76
4.3.3.5.Onuncu Dereceden Süzgeç Kullanılarak AGG:..... 78
4.4. FARKLI SNR DEĞERLERİNDE AGG'NİN BAŞARIMI.....……80
4.5. FG ALGORİTMASININ KONTROL PARAMETRELERİNİN AGG TASARIMINA ETKİSİ…….83

5. BÖLÜM……...87
SONUÇ VE ÖNERİLER 87

KAYNAKLAR……89
EK–1 97

ÖZGEÇMİŞ 101

KISALTMALAR

FGA	Farksal Gelişim Algoritması
AGG	Adaptif Gürültü Giderici
LMSA	En Küçük Ortalama Kareler Algoritması
NLMSA	Normalize Edilmiş En Küçük Kareler Algoritması
RLSA	Özyineli En Küçük Kareler Algoritması
FIR	Sonlu Darbe Yanıtlı Sayısal Süzgeçler
IIR	Sonsuz Darbe Yanıtlı Sayısal Süzgeçler
MSE	Ortalama Karesel Hata
LMS	En Küçük Ortalama Karesel Hata
DSP	Sayısal İşaret İşleme
VLSI	Çok Büyük Aralıklı İntegre
SNR	İşaret Gürültü Oranı
CR	Çaprazlama sabiti
F	Ölçekleme sabiti
K	Kombinasyon faktörü
N	Süzgeç derecesi
D	Boyut
best	En iyi değişken
μ	LMS algoritmasının adım büyüklüğü

TABLOLAR LİSTESİ

Tablo 4.1. LMS Algoritmasının kontrol parametreleri…….……...…….45
Tablo 4.2. NLMS Algoritmasının kontrol parametreleri ..…….……...…...…..46
Tablo 4.3. FG algoritmasının kontrol parametreleri ……..……..……...…..47
Tablo 4.4. FG, LMS ve NLMS algoritmaları ile ikinci dereceden süzgeç için elde edilen katsayılar .....…56
Tablo 4.5. FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayılar …….……..57
Tablo 4.6. FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar ….……..……..…….…….59
Tablo 4.7. FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç için elde edilen katsayılar ……..…….60
Tablo 4.8. FG, LMS ve NLMS algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar……62
Tablo 4.9. $\sin(2\pi t^2/8N^2)$ Bilgi İşareti için farklı süzgeç derecelerinde SNR değerleri……..63
Tablo 4.10. FG, LMS ve NLMS algoritmaları ile ikinci dereceden süzgeç için elde edilen katsayılar……...65
Tablo 4.11. FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayılar……66
Tablo 4.12. FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar……68
Tablo 4.13. FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç için elde edilen katsayılar……69
Tablo 4.14. FG, LMS ve NLMS a algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar……71
Tablo 4.15. $\sin(2\pi t^2/8N^2)+ \sin(2\pi t^2/4N^2)$ Bilgi işareti için farklı süzgeç derecelerinde SNR değerleri…….....72
Tablo 4.16. FG, LMS ve NLMS algoritmaları ile ikinci dereceden süzgeç için elde edilen katsayılar……...73
Tablo 4.17. FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayılar……75

Tablo 4.18. FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar……76
Tablo 4.19. FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç için elde edilen katsayılar……78
Tablo 4.20. FG, LMS ve NLMS algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar……79
Tablo 4.21. $\sin(2\pi t^2/8N^2)+ \sin(2\pi t^2/4N^2)+ \sin(2\pi t^2/2N^2)$ Bilgi işareti için farklı süzgeç derecelerinde SNR değerleri……..80
Tablo 4.22. FG algoritmasında kullanılan diğer kontrol parametreleri……..83
Tablo 4.23. FG algoritmasında kullanılan diğer kontrol parametreleri……..85

ŞEKİLLER LİSTESİ

Şekil 2.1. Adaptif süzgeç için blok diyagram……..…….6

Şekil 2.2. Adaptif süzgeç bileşenleri……..……..7

Şekil 2.3. Enine FIR süzgeç yapısı……...…….9

Şekil 2.4. Direkt-Form IIR süzgeç yapısı……...……..10

Şekil 2.5. Adaptif sistem kimliklendirme……..……..13

Şekil 2.6. Adaptif sistem modelleme……..…….13

Şekil 2.7. Adaptif kestirim…….....14

Şekil 2.8. Gürültü giderme……..…….14

Şekil 2.9. Adaptif Gürültü Giderici…….....17

Şekil 2.10. Adaptif enine süzgecin blok diyagramı……..….22

Şekil 2.11. LMS algoritmasının akış diyagramı…..……..…….23

Şekil 2.12. NLMS algoritmasının akış diyagramı…….……..……..25

Şekil 3.1. Farksal Gelişim algoritmasının akış diyagramı……..…...…..31

Şekil 3.2. FG algoritmasında yeni bir çözümün elde edilmesi…….....33

Şekil 3.3. FG algoritmasının şematik diyagramı…….……..……..……34

Şekil 3.4. DE/rand/1/bin'e göre $u_{i,j,G+1}$'i elde etmek için oluşan süreç ve dışhat(contour) çizgileriyle amaç fonksiyonunun gösterimi…….……...38

Şekil 3.5. D=7, n=3 ve L=3 için çaprazlama örneği……39

Şekil 3.6. DE/best/2/bin'e göre $u_{i,j,G+1}$'i elde etmek için oluşan süreç ve kontür çizgileriyle amaç fonksiyonunun gösterimi……..……...40

Şekil 4.1. LMS-NLMS algoritması kullanılarak adaptif FIR gürültü giderici ……42

Şekil 4.2. FG algoritmasının AGG'de kullanılması…….....43

Şekil 4.3. Bilgi işareti, s(n)……...……44

Şekil 4.4. Gürültü işareti, n0(n)……...…….44

Şekil 4.5. Gürültülü işaret, d(n)……...…….45

Şekil 4.6. Tek bir sinüzoidal işaret için LMS algoritması kullanıldığında AGG çıkışı …….……...46

Şekil 4.7. Tek bir sinüzoidal işaret için NLMS algoritmasında elde edilen AGG çıkışı ……46

Şekil 4.8. Tek bir sinüzoidal işaret için FG algoritmasında elde edilen AGG

çıkışı……..…….47
Şekil 4.9. Bilgi işareti, s(n)……...……48
Şekil 4.10. Gürültü işareti, n0(n)……...……48
Şekil 4.11. Gürültülü işaret, d(n)……...……49
Şekil 4.12. İki sinüzoidal işaret toplamında LMS algoritmasında elde edilen AGG çıkışı …….……...…49
Şekil 4.13. İki sinüzoidal işaret toplamında NLMS algoritmasında elde edilen AGG çıkışı ……....50
Şekil 4.14. İki sinüzoidal işaret toplamında FG algoritmasında elde edilen AGG çıkışı ……..……...…...50
Şekil 4.15. Bilgi işareti, s(n)……...……51
Şekil 4.16. Gürültü işareti, n0(n)……...……52
Şekil 4.17. Gürültülü işaret, d(n)……...……52
Şekil 4.18. Üç sinüzoidal işaret toplamında LMS algoritmasında elde edilen AGG çıkışı …….…...……53
Şekil 4.19. Üç sinüzoidal işaret toplamında NLMS algoritmasında elde edilen AGG çıkışı ……...….53
Şekil 4.20. Üç sinüzoidal işaret toplamında FG algoritmasında elde edilen AGG çıkışı……...…...……...54
Şekil 4.21. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi iki olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı…….55
Şekil 4.22. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi dört olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı…….57
Şekil 4.23. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi altı olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı ...……..…..…….58
Şekil 4.24. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi sekiz olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı ..…….60
Şekil 4.25. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi on olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG

çıkışı ...…….62
Şekil 4.26. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi iki olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı……..….64
Şekil 4.27. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi dört olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….66
Şekil 4.28. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi altı olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….67
Şekil 4.29. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi sekiz olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….69
Şekil 4.30. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi on olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….70
Şekil 4.31. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi iki olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı…….73
Şekil 4.32. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi dört olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….74
Şekil 4.33. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi altı olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….76
Şekil 4.34. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi sekiz olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı …….77
Şekil 4.35. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi on olduğunda (a) FG Algoritmasının (b) LMS Algoritmasının (c) NLMS Algoritmasının AGG çıkışı ...…….79
Şekil 4.36. 10 dB değerindeki gürültü işareti, $n_0(n)$……...…81
Şekil 4.37. SNR değeri 10 dB olduğu zaman AGG çıkışı…….81

Şekil 4.38. 20 dB değerindeki gürültü işareti, $n_0(n)$……..…82
Şekil 4.39. SNR değeri 20 dB olduğu zaman AGG çıkışı…….82
Şekil 4.40. K değeri 0.4 iken AGG çıkışı……..84
Şekil 4.41. K değeri 0.6 iken AGG çıkışı……..84
Şekil 4.42. F değeri 0.5 iken AGG çıkışı……...…85
Şekil 4.43. F değeri 0.8 iken AGG çıkışı……...86

1. BÖLÜM

GİRİŞ

Bu tez çalışmasında Farksal Gelişim (FG-Differential Evolution,) Algoritması kullanılarak adaptif FIR (Finite Impulse Response-Sonlu Darbe Cevaplı) süzgeçlerde gürültü gidermeye yönelik bir algoritma geliştirilmiştir. Klasik bir algoritma olan LMS (Least Mean Square-En Küçük Ortalama Kareler) algoritması ve bu algoritmadan türetilen NLMS (Normalized Least Mean Square-Normalize Edilmiş En Küçük Ortalama Kareler) algoritması kullanılarak algoritmanın işlevselliği araştırılmıştır. Ayrıca gürültü giderme için farklı süzgeç derecelerinde yazılan algoritmanın başarımı ölçülmüştür. Adaptif FIR gürültü gidericinin farklı SNR (Signal to Noise- İşaret Gürültü Oranı) değerleri için başarımı ve tasarımda kullanılan Farksal Gelişim algoritmasının kontrol parametrelerinin değişiminin gürültü gidericiye etkisi incelenmiştir. Yine önerilen algoritmanın işlevselliğini göstermek için farklı süzgeç derecelerine sahip işaretlerin LMS ve NLMS algoritmalarıyla karşılaştırılmaları durumunda tasarlanan gürültü gidericinin başarımı ölçülmüştür.

Süzgeçleme, bir işaretin frekans spektrumunun değiştirilerek tekrar şekillendirildiği veya bazı arzu edilen özel operasyonlara tabii tutulduğu bir işlemdir. Süzgeçleme, genelde ya gürültü gibi işaretteki bozulmaları gidermek veya iki ya da daha fazla işaretin kanal kullanımını daha verimli hale getirmek için karıştırıldığı yerlerde; bu işaretleri tekrar ayırmak, işaretleri frekans elemanlarına bölmek, modüleli bir işaretin demodülasyonunu almak, ayrık zamanlı işaretleri sürekli zaman işaretlerine çevirmek gayeleri için kullanılmaktadır.

Bir taşıt stereosunda motordan kaynaklanan harmoniklerin bastırılması gibi süzgeçleme gerektiren hemen hemen bütün uygulamalarda, gerekli frekans cevabı önceden bilinememektedir ya da zamanla değişen bir işaret olmaktadır. Çoğu uygulamada, kendi kendini otomatik olarak tasarlayabilen ve zamana göre sistem değişikliklerini

izleyebilen adaptif bir süzgeç oldukça yararlı olmaktadır. Adaptif süzgeçler birçok uygulamada özellikle de haberleşme alanında kullanılmaktadır.

Süzgeç tasarımcıları, FIR (Finite Impulse Response-Sonlu Darbe Cevaplı) ve IIR (Infinite Impulse Response-Sonsuz Darbe Cevaplı) süzgeçler gibi farklı birçok süzgeç yapısını, bunların doğrudan ya da kaskat şekillerini veya kafes yapılarını vs. incelemektedirler. Çoğu süzgeç tasarım işleminde tasarımcı, önce bir süzgeç yapısını ardından süzgeç katsayılarını ve sonra da seçilen süzgecin ihtiyaçları yerine getirip getirmediğini belirlemektedir. Bu işlem eğer dikkatlice gerçekleştirilirse, tatmin edici sonuçlar alınabilir. Birçok tasarım algoritması daha az karmaşık yapılar üzerinde yoğunlaştığı için seçilen süzgeç yapısı yine de optimallığı sağlayamamaktadır. Giriş verisinin karakteristiği zamana bağlı olarak değişirse ya da giriş verisi hakkındaki gereken bilgi yeterli olmazsa adaptif süzgeçlere ihtiyaç duyulmaktadır. Adaptif süzgeçlerin tasarlanması için birçok metot vardır. Bilgisayarlar, başarımlarının gelişimi ile evrimsel algoritmalara dayanan optimizasyon (en iyileme) metotlarının yeni bir sınıfını kullanarak bazı teknik problemlerin çözümüne izin vermektedirler. Çeşitli algoritmalar bu problemler için kullanılmış ve yeni algoritmalar da ortaya çıkmıştır. Bunlardan birisi de gelişime dayalı algoritmalardan FG algoritmasıdır [1].

Price ve Storn, FG'nin çalışma prensibini ortaya koymuşlardır [2]. Daha sonraki çalışmalarında, FG algoritması için on farklı strateji önermişlerdir. Yine yaptıkları çalışmalarla FG algoritmasının anahtar parametrelerinin seçimi için çeşitli kurallar belirlemişlerdir[3]. FG algoritması çeşitli alanlarda birçok probleme uygulanmış ve başarılı olmuştur. FG algoritmasının çeşitli uygulamaları aşağıda verilmektedir:

Storn [4], "Sayısal süzgeç tasarımı" hakkında çalışma yaparken, Comon ve Pham [5], "Gürültü giderici için bir FIR süzgecin tahmin derecesi" üzerine çalışma yapmışlardır. Graupe ve Efron [6] ise "Adaptif aktif gürültü giderici için beyaz çıkış (Output-Whitening) yaklaşımı" ile ilgilenmişlerdir. Kwan ve Li [7], "Broadband (geniş bant) gürültü giderici için yeni bir doğrusal olmayan adaptif FIR sayısal süzgeç" tasarlamışlardır. Fah ve arkadaşları [8], "Yapay sinir ağı kullanılarak gürültü gidericiyle ses çoğaltımı" üzerine çalışırken, Kang ve Fransen [9], "Bir adaptif gürültü giderici süzgeçle deneyler" yapmışlardır. Savoji [10], "Adaptif gürültü giderici için değişken

uzunlukta bir kafes süzgeç" tasarlamıştır. Li ve Er [11] "Bulanık sistem tabanlı IIR süzgeç kullanılarak adaptif gürültü giderici" üzerine çalışmışlardır. Xie ve Kwan [12], "Adaptif gürültü giderici için çok kanallı kafes yapısı" konusunu çalışırken, Xie ve Kwan [13] ise "Çoğaltılmış dinamik bulanık yapay sinir ağları kullanılarak adaptif gürültü giderici" tasarlamışlar ve Li ve arkadaşları [14] da "Adaptif gürültü süzgeçleme yaklaşımıyla bilgisayar yazılımı"nı gerçekleştirmişlerdir. Kao ve arkadaşları [15] "Sayısal kamera uygulamaları için karışık örnek radyo dalgalarıyla iletilen adaptif gürültü giderimi", Shimauchi [16] tarafından "Akustik yankı giderimi için güçlü bir NLMS algoritması" tasarlanmış, Efron ve Han [17] "Geniş çapta adaptif aktif gürültü giderici" yi incelemiş ve Mayer ve arkadaşları [18] ise "Model optimizasyonu için FG algoritması" araştırmasını gerçekleştirmişlerdir. Shiakolas ve arkadaşları [19] "Mühendislik Problemlerinde FG'nin kullanılması" konusunu incelerken, Karaboğa ve Çetinkaya [20] "Sabit noktalı sayısal süzgeçlemede FG'nin kullanılması" üzerine çalışma yapmışlardır. Kukkonen [21], "Kontrol parametrelerinin deneysel çalışmasında FG'nin kullanılması" konusu ile ilgilenmiş, Rönkkönen ve arkadaşları [22] "FG ile gerçek parametreli optimizasyon" çalışmasını gerçekleştirmiş ve Kukkonen ve Lampinen'de [23], "FG gelişiminin üçüncü adımı" isimli çalışmayı yapmışlardır. Price ve arkadaşları [24] "Küresel optimizasyona pratik bir yaklaşım olarak FG" konusuyla çalışmalar yaparken, Zielinski ve arkadaşları [25,26] ise "FG ile sınırlı çok amaçlı optimizasyon" ve "FG ve PSO (Particle Swarm Optimization, Parçacık Sürüsü Optimizasyonu) için durdurma kriterine göre yürütüm süresi analizleri" konulu iki çalışmaya imza atmışlardır. Weitkemper ve arkadaşları [27] "FG açısından kodlu CDMA(Coded Division Multiple Access, Kod Bölmeli Çoklu Erişim) sistemlerinde girişim bastırma optimizasyonu" konusu ile ilgili çalışmışlardır. O'neill ve Brabazon [28] "Kurallı FG" üzerine, Robic ve Filipic'de [29] "Çok amaçlı optimizasyon için FG" tasarımı üzerine çalışma yapmışlardır. Yine Braak ve Ter'in [30] "Genetik algoritma FG'nin Marcov zinciri Monte Carlo versiyonu" konulu çalışması ve Besson ve arkadaşlarının [31] "FG ile kuramlı optimizasyon problemi eğitim metodu" konulu çalışmaları ses getirirken Karaboğa ve Çetinkaya'nın [32] "Adaptif FIR süzgeç tabanlı Genetik algoritma kullanılarak gürültü giderme" konulu çalışması FG algoritması uygulamaları için yol gösterici birer örnek olarak verilebilir.

Gerçekleştirilen tez çalışmasında, adaptif FIR süzgeçlerde gürültü giderme problemi

için, optimizasyon algoritmalarından FG algoritmasına dayalı bir yapı önerilmiştir. Tasarlanan algoritmanın başarımını ölçmek için literatürde bulunan ve oldukça sık kullanılan klasik algoritmalardan LMS ve NLMS algoritmaları kullanılmıştır. Yine farklı süzgeç derecelerinde, gürültü gidermenin başarımları üzerine karşılaştırmalı çalışmalar yapılmıştır.

Yapılan tez çalışması, beş bölüm olarak planlanmıştır. Birinci bölümde tezin konusu ve literatür araştırması verilmiştir. İkincisi bölümde, adaptif süzgeçler ve adaptif FIR süzgeçlerde gürültü giderme konusu hakkında bilgi verilmiş olup gürültü gidermede literatürde oldukça sık olarak kullanılan, klasik teknikler olarak bilinen LMS ve NLMS algoritmaları açıklanmıştır. Üçüncü bölümde, yapay zeka optimizasyon algoritması olan FG algoritması detaylı olarak açıklanmıştır. Dördüncü bölümde, adaptif FIR süzgeçlerde gürültü giderme problemi için önerilen FG algoritması ile elde edilen sonuçlar sunulmuştur. Tasarlanan algoritmanın başarımını ölçmek için, klasik algoritmalardan LMS ve NLMS algoritmaları kullanıldığında elde edilen sonuçlarda, bu bölümde verilmiştir. Farklı süzgeç derecelerinde, gürültü gidermenin başarımları üzerine karşılaştırmalı çalışmalara ek olarak, farklı SNR değerlerinde algoritmanın başarımı incelenmiştir. Bu bölümün son kısmında ise, FG algoritmasının kontrol parametreleri olan K (Combination Factor-Birleşme Faktörü) ve F (Scaling Factor-Ölçekleme Faktörü) değerlerinin değişiminin, AGG üzerindeki etkisi araştırılmıştır. Beşinci bölümde, elde edilen bulguların karşılaştırılması ile varılan sonuçlar açıklanmıştır.

2. BÖLÜM

ADAPTİF SÜZGEÇLER

Adaptif süzgeçleme, teorileri ve uygulamaları 1950'lerin sonlarına doğru ortaya çıkmıştır. Widrow ve arkadaşlarının bu alanda yaptığı çalışmalar 1959'da ünlü LMS algoritmasının bulunması ile sonuçlanmıştır [1]. Öncelikle bu alan, 1970'lerin ortalarındaki sayısal teknolojinin ilerlemesiyle popüler olmuştur. Adaptif süzgeçleme, haberleşme, ses işleme, görüntü işleme, gürültü ve yankı giderme gibi alanlarda başarılı bir şekilde uygulanmaktadır. Adaptif süzgeçler, haberleşme, radar, kontrol, sonar, sismoloji, görüntü işleme ve model tanıma gibi birçok farklı alanda başarılı bir şekilde kullanılmaktadırlar. Sistem kimliklendirme, kanal dengeleme, ses tanıma, gürültü giderme, işaret bulma, spektrum analizi ve hat çoğaltma adaptif süzgeçlerin tipik uygulamalarıdır [3].

Bu uygulamalar karakteristikleri önceden bilinmeyen sistemlerin ürettiği işaretleri işleyen süreçleri içermektedir. Bu şartlar altında, sabit katsayılı süzgeçlerin yerine adaptif süzgeçlerin kullanılmasıyla çok önemli bir başarım elde edilmektedir. Adaptif sistemler çevre şartlarına göre kendi kendisini en iyiye doğru kanalize eden sistemlerdir. Bu sistemler en çok zamanla değişen işaret ve sistemlerde uygulanmaktadırlar. Adaptif sistemlerin en önemli özelliği, zamanla değişen sistemlere rahatlıkla uygulanabilmesi ve yeni durumlara göre kendi kendini ayarlayabilmesidir [4].

2.1. Adaptif Süzgeçler

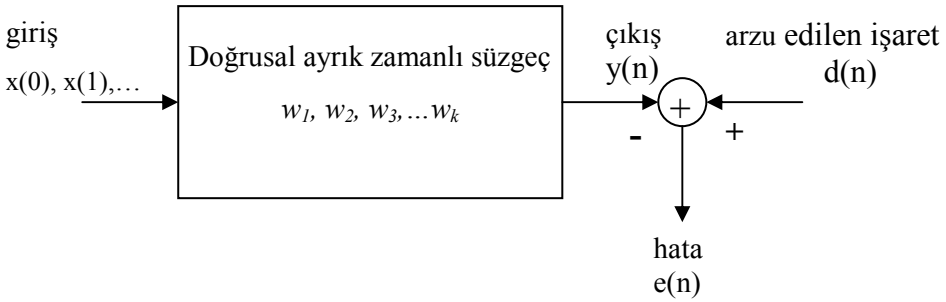
Normal sabit bir süzgeç işaretin ve istenmeyen gürültünün her ikisinin de istatistiklerinin önceden bilinmesiyle tasarlanmaktadır. Oysa adaptif süzgeç recursive (yinelemeli) algoritmalar vasıtasıyla değişen çevre parametrelerine göre sürekli ayar yapan bir süzgeçtir. Adaptif süzgeçler, zamanla değişen ve önceki değerleri bilinmeyen

işaretler için tasarlanmıştır.

Adaptif süzgeçler sayısal, analog ve karışık olarak tasarlanabilmektedirler. Her tekniğin kendine göre avantajı ve dezavantajı vardır. Örneğin analog adaptif süzgeçler, çok hızlı fakat offset'i en küçük hatayı elde etmekte zorlanır. Sayısal süzgeçler yavaştır fakat kararlıdır. Analog ve sayısal tasarımın birlikte kullanıldığı karışık tasarımlar kararlılık ve hız açısından iyi bir seçenek sunmaktadırlar. Fakat VLSI (Very Large Scale Integration- Çok Büyük Aralıklı Entegre) tasarımları çip içerisinde analog ve sayısal bileşenleri ayırmak gerekli olduğu için daha karışıktır.

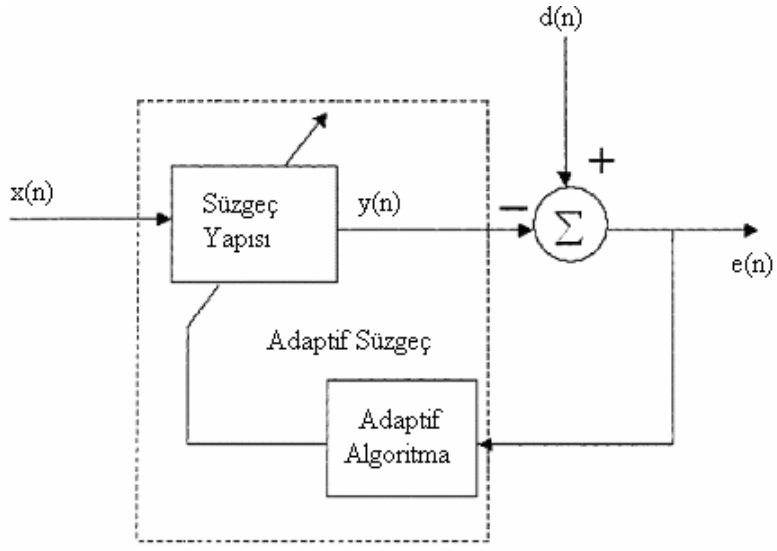
Bir adaptif süzgeç, bir maliyet fonksiyonunu optimize etmek için veya önceden belirlenen optimizasyon kriterlerini karşılamak için katsayıları ayarlanan ve zamanla değişen bir süzgeçtir. Bir adaptif süzgeçleme sistemi, çevresiyle ilişkili olarak davranışını veya başarımını geliştirme şeklinde yapısı düzenlenebilen ve değiştirilebilen bir sistemdir [2]. Şekil 2.1'de bir adaptif süzgecin blok diyagramı verilmektedir.

Şekil 2.1'den de görüldüğü gibi adaptif süzgeçlerde giriş işaret x(n) ve arzu edilen işaret olan referans giriş d(n) olmak üzere iki giriş gerekmektedir ve süzgeç katsayıları w_k'lar vasıtasıyla üretilen bir de y(n) çıkışı vardır. Arzu edilen işaret tahmini hata e(n)'i elde etmek için süzgeç çıkışındaki işaretle karşılaştırılır. Bu hata işareti yeni zaman sabitine göre süzgeç ağırlıklarını ayarlamakta kullanmaktadır [5].



Şekil 2.1. Adaptif süzgeç için blok diyagram.

Bir adaptif süzgeç Şekil 2.2'de gösterildiği gibi programlanabilir bir süzgeç ve adaptif bir algoritmadan oluşmaktadır ve altı ana bileşene sahiptir.



Şekil 2.2. Adaptif süzgecin bileşenleri.

x(n) giriş, y(n) çıkış, d(n) istenen ve e(n) hata işareti olmak üzere ilk dört bileşeni oluştururken süzgeç yapısı ve adaptif işlem kriterleriyle belirlenen adaptif algoritma ile altı ana bileşen tamamlanmış olmaktadır.

Süzgeç için giriş işareti x(n) bir bilgi dizisidir. Süzgeç katsayıları w_k'lar vasıtasıyla üretilen bir de y(n) çıkışı vardır. Süzgeç çıkış dizisi $\hat{y}(n)$, istenen işaret y(n)'in tahmini değeri olduğundan çıkış veya tahmini hata,

$$e(n) = y(n) - \hat{y}(n) \tag{2.1}$$

olur. Giriş işareti x(n) ve çıkış işareti e(n)'e göre, adaptif algoritma bir maliyet fonksiyonunu minimize etmek üzere süzgecin ağırlıklarını yani parametrelerini güncellemektedir. Adaptif süzgeçlerin zamanla değişen parametrelerinden dolayı zamanla değişen bir sistem olduğu söylenebilmektedir. Adaptif süzgeçlerin parametreleri, süzgecin girişine de bağlı olduğundan aynı zamanda doğrusal olmayan bir sistem olarak da bilinmektedir.

Şekil 2.2'den de görüldüğü gibi adaptif süzgeçlerde giriş işareti x(n) ve arzu edilen işaret olan referans girişi d(n) olmak üzere iki giriş gerekmektedir ve arzu edilen işaret tahmini hata işareti e(n)'i elde etmek için süzgeç çıkışındaki işaretle karşılaştırılmaktadır. Bu hata işareti, yeni zaman sabitine göre süzgeç ağırlıklarını

ayarlamakta kullanılmaktadır. Bir x(n) giriş işareti her iterasyonda, yine y(n) çıkışını üretmek için zamanla değişen bir süzgeç tarafından işlenmektedir. Bu işaret, e(n) hata işaretini oluşturmak için d(n) istenen işareti ile karşılaştırılmaktadır. İstenen işaret d(n) ile çıkış işareti y(n) arasındaki fark olan bu hata işareti e(n), verilen başarım kriterini veya maliyet fonksiyonunu minimize eden adaptif süzgeç katsayılarını ayarlamak için adaptif algoritmaya geri beslenmektedir. Maliyet fonksiyonunu minimize etmek demek, farklı uygulamalarda bazı anlamlarda adaptif süzgeç çıkış işaretinin istenen çıkış işaretiyle uyumlu olduğu anlamına gelmektedir. Ağırlıkları ayarlamak amacıyla kullanılan birçok algoritma mevcuttur. Eğitim algoritmasının seçimi yakınsama hızına ve hesaplama karmaşıklığına bağlıdır [2].

Maliyet Fonksiyonu: Maliyet fonksiyonu hatanın ölçümü olarak bilinmektedir. Aşağıda genellikle kullanılan iki farklı maliyet fonksiyonu verilmektedir.

(1) Ortalama Karesel Hata (Mean Square Error-MSE); çıkış hatasının gücü, ortalama karesel hata olarak bilinmektedir.

$$\varepsilon(n) = E\left\{e^2(n)\right\} \tag{2.2}$$

(2) Ağırlıklı Karesel Hatanın Toplamı (The Sum of Weighted Square Error); çıkış hatasının pencereleme enerjisi, ağırlıklı karesel hata olarak bilinmektedir.

$$\varepsilon(n) = \sum e^2(i) w^{n-i} \qquad 0 < w \leq 1 \tag{2.3}$$

2.1.1. Süzgeç Yapısı

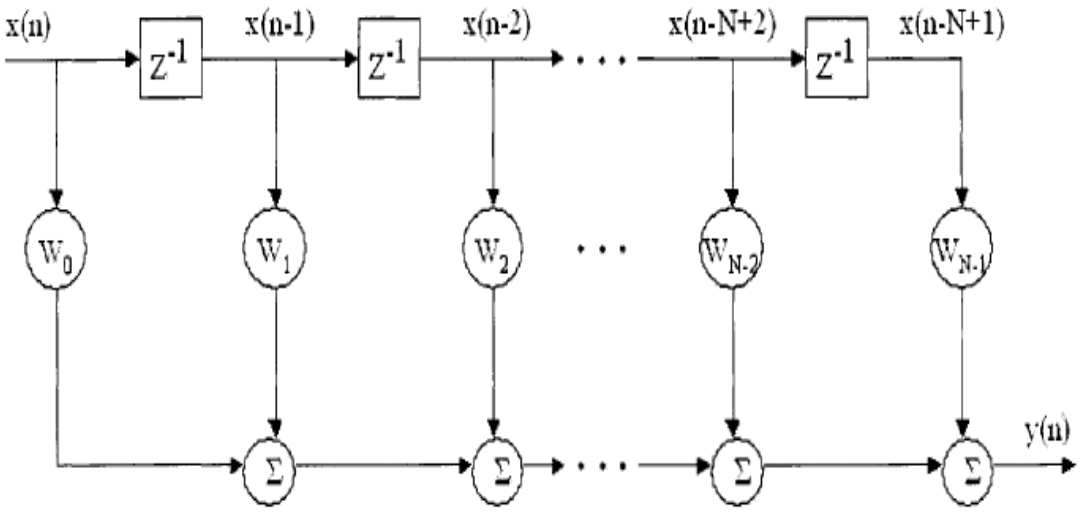
Temel olarak, adaptif süzgeçlerin gerçekleştirilmesinde iki ana yapı vardır; FIR ve IIR süzgeç yapıları [2].

2.1.1.1. FIR Süzgeç Yapısı

FIR üzerine kurulan adaptif süzgeç yapısı pratik uygulamalarda tercih edilmektedir. Adaptif FIR süzgeçlerin en büyük dezavantajı belirli uygulamalarda iyi bir başarım elde etmek için çok büyük parametreler gereksinim duymasıdır ki bu da hesaplama maliyetini arttırmaktadır.

En çok kullanılan adaptif FIR süzgeç enine (transversal) süzgeçtir ve Şekil 2.3'de gösterildiği gibi vuruş gecikmeli hat olarak da adlandırılmaktadır [3]. Şekil 2.3'de verilen x(n), giriş işareti, y(n) çıkış işareti ve w_0, w_1, w_3,….,w_{N-1} ise süzgeç katsayılarıdır. Süzgeçte kullanılan gecikme elemanı sayısı, darbe cevabının sonlu süresini belirlemektedir. Şekilde n-1 ile gösterilen gecikme elemanı sayısı genellikle süzgeç derecesi olarak belirlenir. Bu şekilde gecikme elemanlarının her biri y^{-1} birim gecikme operatörüyle belirtilmiştir. Özel olarak d^{-1}, u(n) birim basamak girişine uygulanırsa, çıkış u(n-1) dir. Bir FIR süzgeç her zaman kararlıdır. FIR süzgeçler nonrecursive (yinelemesiz) süzgeçler olarak da adlandırılmaktadır ve yinelemesiz fark denklemi Eşitlik (2.4)'deki gibi gösterilebilir.

$$\hat{y}(n) = \sum_{j=0}^{N} w_j(n)x(n-j) \tag{2.4}$$



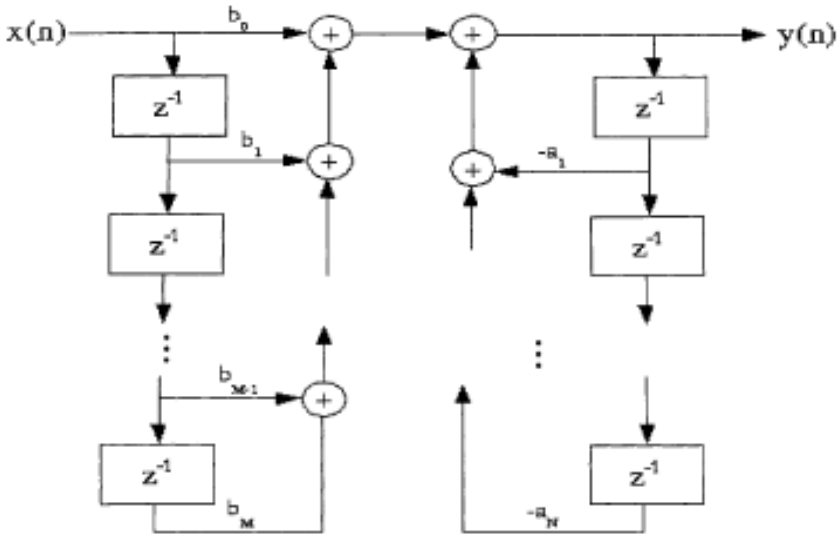
Şekil 2.3. Enine FIR süzgeç yapısı.

2.1.1.2. IIR Süzgeç Yapısı

Adaptif IIR süzgeçleri gerçekleştirmek için en yaygın kullanılan, uygulama ve analizinin kolay olması sebebiyle kanonik yapısıdır. Kafes [9-10], kaskat [11] ve paralel [12] gibi farklı yapılarda araştırmacılar tarafından çalışılmaktadır [2]. Şekil 2.4'de direkt form IIR süzgeç yapısının blok diyagramı gösterilmektedir. Şekil 2.4'de verilen FIR süzgeç için x(n), giriş işareti, y(n) çıkış işareti ve z^{-1} birim gecikmeyi temsil etmektedir. Bir IIR süzgecin çıkışı Eşitlik (2.5) ile verilmektedir.

$$\hat{y}(n) = \sum_{i=1}^{N} a_i(n)\,\hat{y}(n-i) + \sum_{j=0}^{M} b_j(n)x(n-j) \tag{2.5}$$

IIR süzgeçler yinelemeli süzgeçler olarak da adlandırılmaktadır.



Şekil 2.4. Direkt-form IIR süzgeç yapısı.

IIR tabanlı adaptif süzgeç yapısı sıfır-kutup model yaklaşımına sahiptir. Adaptif IIR süzgeçler en az katsayı ile tasarlanabilmektedir. IIR süzgeçlerde kararsızlık problemiyle karşılaşılabilmektedir.

2.1.2. Adaptif Süzgeçlerin Karakteristikleri

Adaptif süzgeçlerin karakteristikleri aşağıda maddeler halinde verilmektedir:

- Değişen çevre ve sistem istekleriyle karşılaştıklarında otomatik olarak kendilerini ayarlayabilirler [2].
- Bazı problemlere göre özel süzgeçler yapmaya ve karar almaya yönelik eğitilebilirler.
- Genellikle zamanla değişen parametreleriyle, doğrusal olmayan bir sistem olarak tanımlanabilirler.
- Genellikle adaptif olmayan süzgeçlerden daha karışıktırlar ve daha zor analiz edilmektedir. Fakat giriş işaretleri bilinmediği veya zamanla değiştiğinde sistem başarımını arttırmak için farklı olasılıklar sunarlar.

2.1.3. Adaptif Algoritma

a) **Yakınsama Oranı:** Yakınsama oranı, algoritmanın durağan girişlere cevabının ortalama kare duyarlılığında optimum Wiener çözümüne yeterince yaklaşması için gereken iterasyon sayısı olarak tanımlanır. Yani sabit şartlar altında, optimuma "yeteri kadar yakın" yakınsamak amacıyla algoritma için gerekli iterasyonların sayıdır.

b) **Yanlış Ayar (Misadjustment):** Bu parametre, Wiener süzgeç tarafından üretilen minimum ortalama karesel hatadan türetilen ortalama karesel hatanın son değerinin nicel bir ölçümünü üretir.

c) **İzleme:** Bir adaptif süzgeçleme algoritması, durağan olmayan bir çevrede çalıştığı zaman çevredeki istatistik değişiklikleri izlemesi gerekir. Algoritmanın izleme başarımı iki özellikten etkilenir. Bunlardan birincisi yakınsama oranıdır. İkincisi ise algoritma gürültüsü nedeniyle sürekli durum dalgalanmasıdır.

d) **Güçlülük (Robustness):** Sonlu kelime uzunluğunu kullanarak çalışan sayısal bir bilgisayar üzerinde adaptif süzgeçleme algoritmalarının gerçekleştirilmesi kuantalama hatalarıyla sonuçlanmaktadır. Bu hatalar bazen adaptasyon algoritmasının sayısal kararsızlığına neden olabilmektedir. Bir adaptif süzgeçleme algoritmasının sonlu kelime uzunluklu işlemlerini kullanan sayısal uygulamalarının sabit olduğu durumlarda nümerik olarak güçlü(robust) olması beklenir.

e) **Hesaplama İhtiyaçları:** Bu pratik açıdan önemli bir parametredir. İlgili konular: (a) algoritmanın iterasyonlarını eksiksiz yapmak için işlemlerin sayısı, (b) veri ve programı depolamak için hafıza miktarı, (c) bir bilgisayarda algoritmayı programlamak için gereken maliyet olarak sayılabilir.

f) **Yapı:** Bu donanım şeklinde yerine getirilen, tarzı belirleyen algoritmada bilgi akışının yapısını göstermektedir.

g) **Sayısal Özellikler:** Bir algoritma sayısal olarak çalıştığında ortaya çıkan yanlışlıklar, kuantalama hatalarından kaynaklanmaktadır.

Standford Üniversitesinde Widrow ve arkadaşlarının LMS algoritmasıyla çalıştıklarından sonraki birkaç yılda çeşitli algoritmalar geliştirilmiştir [6]. Bu

algoritmalar iki kategoriye ayrılır: Wiener süzgeç teorisi tabanlı LMS algoritma ailesi ve en küçük kareler teorisi ve Kalman süzgeç tabanlı RLS (Recursive Least Squares – Yinelemeli En Küçük Kareler) algoritma ailesi. Bütün bu algoritmalar arasında, NLMS algoritması basitliği ve keskinliği yüzünden çok geniş çapta uygulama alanı bulmuştur.

Adaptif bir süzgecin ağırlıklarını yenilemek için birçok algoritma mevcuttur. LMS algoritması en yaygın kullanılan algoritmadır. Şekil 2.2'de verilen bir adaptif süzgeç gösteriminden amaç; $e(n) = d(n) - y(n)$ eşitliğini minimize edecek şekilde hata işaretine bağlı bir algoritma fonksiyonuyla doğrusal süzgeci tasarlamaktır. İyi bilinen Wiener süzgeçleme problemi için, amaç fonksiyonu E[e^2(n)]'dir [13]. Çıkış işareti *y(n)* Eşitlik (2.6)'da verildiği gibi hesaplanmaktadır:

$$y(n) = W_n^T X_n \tag{2.6}$$

Burada,

$$W_n = [w_n(0), w_n(1), ..., w_n(N-1),]^T \tag{2.7}$$

W_n n. iterasyonda FIR süzgecin katsayı vektörüdür ve *N* süzgeç uzunluğudur.

$$X_n = [x_n, x_{n-1}, x_{n-2}, ..., x_{n-N+1}] \tag{2.8}$$

Xn , n. iterasyonda adaptif süzgecin giriş vektörüdür.

LMS algoritması W_n katsayılarını yenileyen en basit algoritmadır. Bu algoritma süzgeç katsayılarını aşağıdaki gibi yeniler:

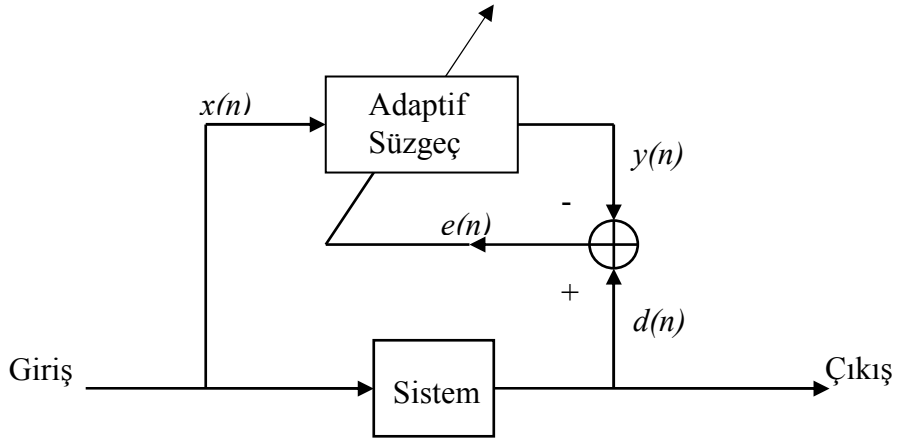
$$W_{n+1} = W_n + \mu e(n) X_n \tag{2.9}$$

μ adım büyüklüğüdür ve μ >0 olmalıdır. Amaç her bir iterasyonda süzgeç katsayılarını elde etmektir.

2.1.4. Adaptif Süzgeçleme Uygulamaları

Adaptif süzgeçleme uygulamaları, haberleşme sistemleri, kontrol sistemleri, sismoloji, biyomedikal elektronik ve süzgeçleme gibi geniş bir spektrumu kapsamaktadır. Adaptif süzgeçleme uygulamaları temelde dört sınıfa ayrılmaktadır [14];

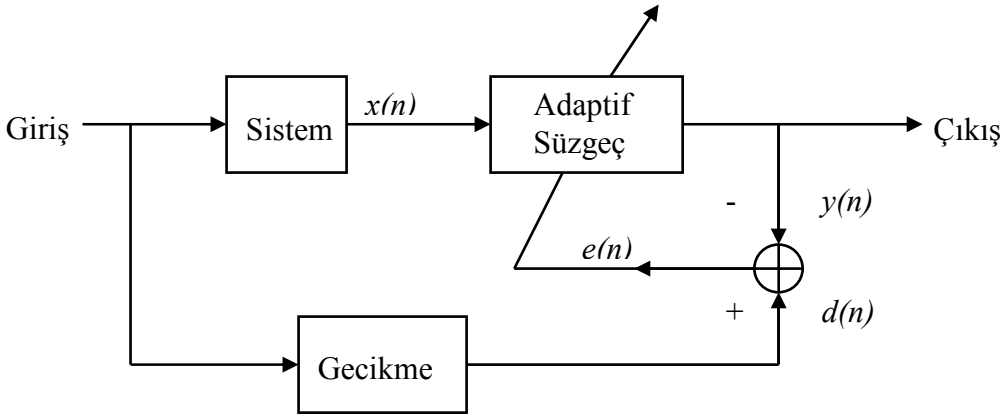
- **Sistem Kimliklendirme:** Adaptif süreç hata işaretinin gücünü indirgemeye çalışırken bilinmeyen sistemin darbe cevabını tahmin eden bir yöntemle süzgeç katsayıları ayarlanmaktadır.



Şekil 2.5. Adaptif sistem kimliklendirme.

Bir adaptif süzgeç bilinmeyen bir sistemin karakteristiklerini tahmin etmek için bir model olarak kullanılır. Şekil 2.5'de adaptif sistem kimliklendirmenin blok diyagramı verilmektedir. Burada x(n) giriş, y(n) çıkış, d(n) istenen ve e(n) hata işaretidir.

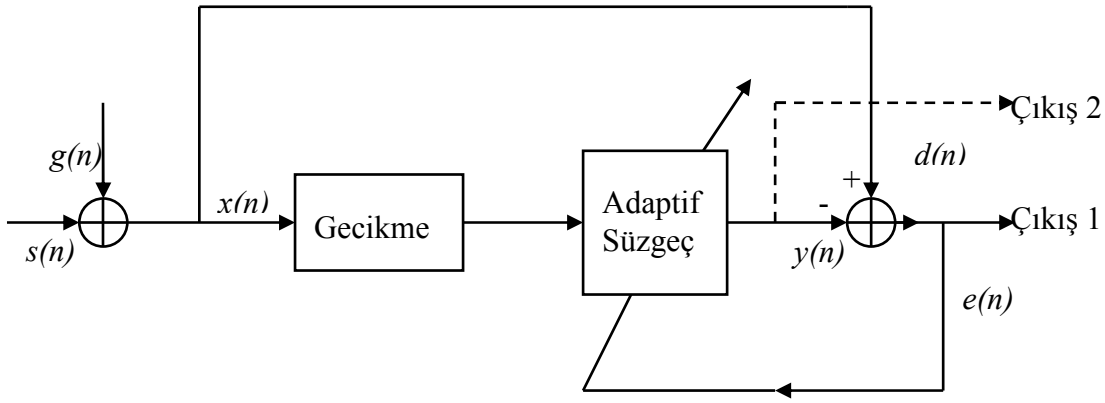
- **Sistem Modelleme:** Sistem modellemede, Kimliklendirmenin tersine, ters darbe cevabı tahmin edilmektedir. Şekil 2.5 aynı zamanda adaptif sistem modelleme blok diyagramı olarakda incelenilebilir. Şekil 2.6'da ise adaptif ters modellemenin blok diyagramı gösterilirken, x(n) giriş, y(n) çıkış, d(n) istenen ve e(n) hata işaretini temsil etmektedir.



Şekil 2.6. Adaptif ters modelleme.

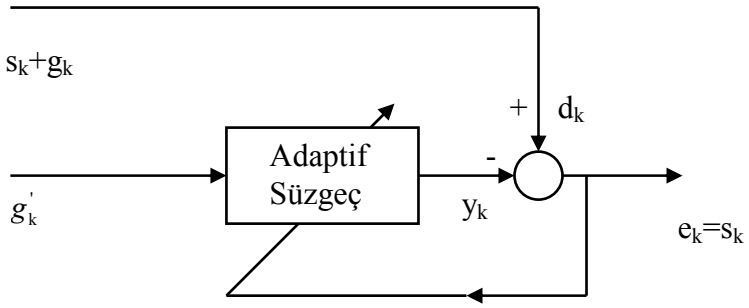
- **Kestirim:** Gürültü g(n) periyodik işaret s(n)'e eklenmiştir. Adaptif süzgecin

y(n) çıkışı s(n)'nin optimal tahminidir. Hata işareti e(n), gürültü işareti g(n)'nin bir tahminidir. Şekil 2.7'de adaptif kestirimin blok diyagramı verilmektedir.



Şekil 2.7. Adaptif kestirim.

- **Gürültü Giderme:** Bir adaptif süzgeçte istenen işaretteki gürültüyü elimine etmek ve tahmin etmek için kullanılır. Referans işaret s_k işareti ile eklenebilir g_k gürültüsünün toplamı ve giriş işareti de gürültü işareti g_k ile arasında korelasyon olan g_k işaretidir. Adaptif süzgeç gürültü işareti g_k'ya benzeyen bir y_k çıkış işareti üretecektir. Toplam çıkış e_k işaret s_k'nın optimal bir tahminidir. Şekil 2.8'de adaptif gürültü gidermenin blok diyagramı verilmektedir.



Şekil 2.8. Gürültü giderme.

Süzgeç yapısının FIR ya da IIR mı olacağı, ne tür algoritma kullanılacağı ve öğrenme oranının ne olacağı uygulamaya bağlı olarak değişmektedir. Bu çalışmada, FG algoritması ile adaptif FIR süzgeç tasarlanması ve gürültü giderme üzerine odaklanılmıştır.

2.2. Adaptif Gürültü Giderme

Eklenebilir gürültüyle bozulmuş bir işareti tahmin etmenin en genel yolu, nispeten değişmemiş işarete izin verirken gürültüyü bastırmaya yönelik bir süzgeç vasıtasıyla birleşik işareti geçirmektir. Böyle süzgeçlerin tasarımı, Wiener ile başlayan ve Kalman, Buce ve diğerleri ile geliştirilen optimal süzgeçleme alanına girmektedir. Belirtilen amaçlar için kullanılan süzgeçler sabit ya da adaptif olabilirler. Sabit süzgeçlerin tasarımı, hem işaretin hem de gürültünün önceki bilgilerine dayalı olmalıdır fakat adaptif süzgeçler, kendi parametrelerini otomatik olarak ayarlama kabiliyetine sahiptir ve tasarımları işaretin ya da gürültünün özelliklerinin önceki bilgilerine çok az ihtiyaç duyularak veya hiç gerek duyulmaksızın yapılmaktadır [1].

Gürültü giderme birçok uygulamada çok fazla avantaja sahip olan optimal süzmenin bir çeşididir. İşaretin algılanamadığı ya da zayıf olduğu gürültü alanındaki noktalara yerleştirilen bir veya daha fazla sensörden elde edilmiş yardımcı ya da referans giriş kullanılır. Giriş süzülür ve işaret ve gürültünün her ikisini de içeren ana (primary) girişten çıkartılır. Sonuç olarak, gürültü zayıflatılır ve giderici tarafından elimine edilir. İlk bakışta, alınan bir işaretten gürültüyü çıkarmak sakıncalı görünmektedir. Eğer uygunsuz şekilde yapılırsa çıkış gürültü gücünde bir artışla sonuçlanmaktadır. Bununla birlikte, eğer süzme ve çıkarma uygun bir adaptif süzgeç tarafından kontrol edilirse, birçok durumda işaretin bozulması veya çıkış gürültü seviyesinin artması gibi küçük risklerle gürültü indirgeme uygulaması çoğu durumda başarılabilmektedir. Adaptif gürültü gidermenin uygulanabildiği durumlarda, direkt süzmeyle başarılması zor veya imkansız olan gürültü giderimi bir derecede başarılabilir.

Adaptif gürültü gidermedeki ilk çalışmalar 1957 ile 1960 yılları arasında General Electric Company'de çalışan Howells, Applebaum ve arkadaşları tarafından yapılmıştır. Yardımcı bir anten ve iki ağırlıklı basit bir adaptif süzgeçten türetilen referans girişi kullanan anten yan kulak (sidelobe) giderimi için bir sistem oluşturmuşlardır [1].

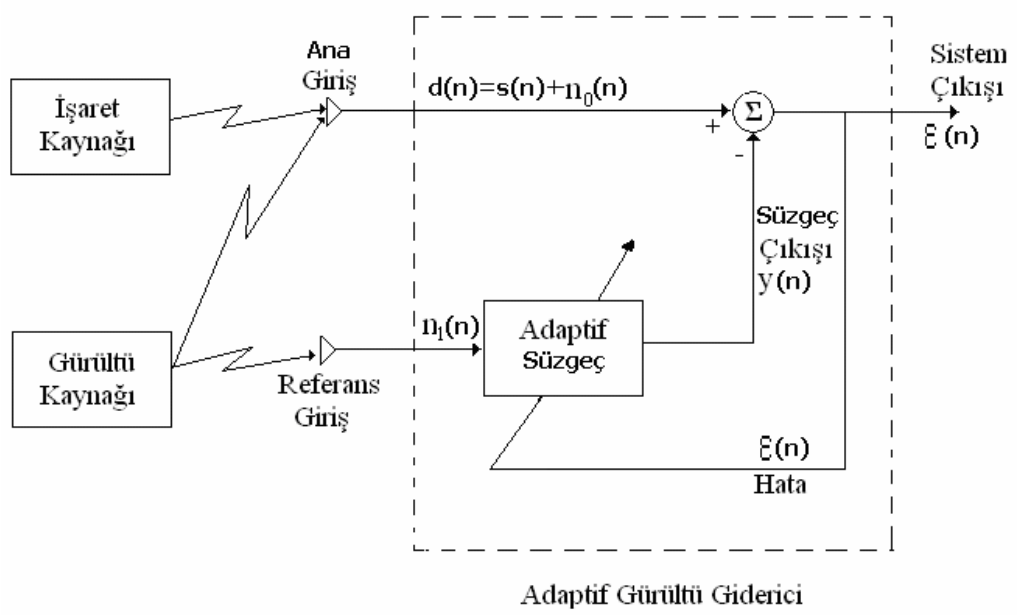
Bu ilk çalışma zamanında, çok az araştırmacı adaptif sistemlerle ilgilenmekteydi. Çok ağırlıklı adaptif süzgeçlerin gelişimi henüz başlamıştı. 1959'da Stanford üniversitesinde Widrow ve Hoff LMS adaptif algoritmasını ve Adaline olarak bilinen örnek tanımlama (pattern recognition) şemasını, adaptif doğrusal eşik lojik elementi için

bulmuşlardır[46]. Rosenblatt, Cornell Aeronautical laboratuarlarında Perceptron'unu oluştururken Aizerman ve onun meslektaşları, USSR, Moskova'da Otomatikler ve Telemekanikler Enstitüsünde makineleri araştıran otomatik bir gradyan (türev) oluşturuyorlardı. Büyük Britanya'da, Gabor ve meslektaşları adaptif süzgeçleri geliştiriyorlardı [46]. Bu çabaların her biri bağımsız şekilde ilerliyordu. 1960'ların başlarında ve ortasında, adaptif sistemler üzerindeki çalışmalar yoğunlaştı. Literatürde adaptasyon, adaptif kontrol birimi, adaptif süzme ve adaptif işaret işleme üzerine yüzlerce makale çıkmaya başladı. Sayısal haberleşmede, adaptif süzgecin önemli bir ticari uygulaması Bell Laboratuarlarında Lucky'nin bu süre boyunca olan çalışmalarından ortaya çıktı [1].

1965'de Stanford üniversitesinde bir adaptif gürültü giderme sistemi gerçekleştirildi. Amacı, elektrokardiyografik kuvvetlendirici ve kaydedicinin çıkışındaki 60 Hertz'lik karışımı gidermekti. İki ağırlıklı bir analog adaptif süzgecin kullanımıyla yapılan sistemin tanımlaması, bugünkü bilgisayar uygulamalarıyla elde edilen analizler ve sonuçlar sunulmuştur [2]. 1965'ten beri adaptif gürültü giderme, genelde periyodik girişim çıkarılması ve uzak mesafe telefon iletim hatlarındaki yankıların çıkarılmasıyla, elektrokardiyografinin diğer yönlerini de içeren bir seri ek problemde başarılı bir şekilde uygulanmıştır. Riegler ve Compton tarafından adaptif antenler üzerindeki bir çalışma Howell ve Applebaum tarafından orijinal olarak yerine getirilen çalışmalarca genelleştirilmiştir [1]. Bu iki büyük çalışma, farklı uygulamalar için tasarlanmış olmasına rağmen 1975'de Widrow tarafından ele alınan Adaptif Gürültü Gidericinin temeli olarak görülmüştür.

2.2.1. Adaptif Gürültü Giderici

Güvenilir gürültü giderim metotları ses işaretinden gürültüyü etkili bir şekilde ayırabilmektedir. Gürültü karakteristikleri olarak işaretin önceki durumunun bilinmemesi ve zamanla değişebilmesi sayılabilmektedir. Bu gürültü karakteristiklerini izlemeye ve tanımaya yatkın adaptif sistemlerin kullanılması önemli olmaktadır. Gürültü giderimi için adaptif süzgeçleme uygulamaları geniş bir alanda kullanılmaktadır.



Şekil 2.9. Adaptif gürültü giderici.

Adaptif gürültü gidericinin blok diyagramı Şekil 2.9'da gösterilmiştir [2]. Temel olarak bir adaptif gürültü giderici ana (primary yada istenen) ve referans adı verilen iki girişe sahiptir.

1. $n_0(n)$ gürültüsüyle bozulmuş bir s(n) bilgi taşıyan işareti içeren ana işaretin, ayrık zamanlı ve örnekleme periyodunun T=1 olduğu varsayılırsa aşağıdaki gibi yazılabilmektedir.

$$d(n) = s(n) + n_0(n) \tag{2.10}$$

s(n) işareti ve $n_0(n)$ gürültüsü birbirleriyle ilişkisizdir. k=0,1,2 ,…N-1 olduğunda

$$E\left[s(n)\mathrm{n}_0(\mathrm{n}-\mathrm{k})\right] = 0 \tag{2.11}$$

dır.

2. Bir şekilde $n_0(n)$ geçmiş gürültüsüyle ilişkili, fakat s(n) işaretiyle ilişkisiz $n_1(n)$ gürültüsünü içeren diğer (gürültü referans işareti) giriştir. Öyle ki, bütün k'lar için

$$E\left[s(n)\mathrm{n}_1(\mathrm{n}-\mathrm{k})\right] = 0 \tag{2.12}$$

ve

$$E\left[n_0(n)\mathrm{n}_1(\mathrm{n}-\mathrm{k})\right] = p(k) \tag{2.13}$$

dır. Burada p(k), k için bilinmeyen bir çapraz ilişkiye sahiptir.

x(n) referans girişi, çıkış işaretini üreten bir adaptif süzgeçle işlenir.

$$y(n) = \sum_{k=0}^{N-1} w_k(n) x(n-k) \tag{2.14}$$

burada $w_k(n)$, adaptif süzgecin ayarlanabilen ağırlıkları ve N, süzgecin derecesidir.

Sistem iki gürültü işareti $n_0(n)$ ve $n_1(n)$ arasındaki ilişkiyi kullanarak ana girişteki gürültü etkisini azaltmaktadır. y(n) süzgeç çıkışı, d(n) ana giriş işaretinden çıkartılırsa elde edilen hata işareti aşağıdaki gibi tanımlanmaktadır.

$$\varepsilon(n) = d(n) - y(n) \tag{2.15}$$

Eğer, ana ve referans sensörlerine iletilen gürültü üzerindeki kanalların özellikleri bilinseydi, genelde n_1'i $y=n_0$'a değiştirebilen sabit bir süzgeç tasarlanabilirdi. Böylece süzgeç çıkışı, ana girişten çıkarılarak sistem çıkışı tek bir işaret olarak elde edilebilirdi. Bununla birlikte, iletim yollarının özellikleri bilinemediği ya da sadece yaklaşık olarak bilinebildiği ve sabit bir doğası olmadığı için sabit süzgeçlerin kullanımı mümkün olmamaktadır. Üstüne üstlük, eğer sabit süzgeç uygulanabilir olmuş olsaydı bile, bu süzgecin karakteristikleri sonucu başarmak için zor ayarlanmış olacaktı ve zayıf hata, çıkış gürültü gücünde artışla sonuçlanabilirdi.

Şekil 2.9'da gösterilen sistemde referans giriş, süzgeç çıkışı üzerinde, bağımlı bir hata işaretine cevap veren LMS algoritması vasıtasıyla, kendi darbe cevabını otomatik olarak ayarlayan adaptif bir süzgeç ile işlenmektedir. Böylece uygun algoritmayla süzgeç, değişen şartlar altında çalışabilmekte ve hata işaretini minimize etmek için kendi kendini sürekli olarak yeniden ayarlayabilmektedir.

Adaptif süreçte kullanılan hata işareti uygulamanın doğasına bağlıdır. Gürültü giderme sistemlerinde, pratik amaç sistem çıkışı $s+n_0-y$'yi üretmektir. Bu amaç, sistem çıkışını adaptif süzgece geri besleyerek ve tüm sistem çıkış gücünü küçültmek için adaptif bir algoritma üzerinden süzgeci ayarlayarak başarılmaktadır. Diğer bir deyişle, bir adaptif gürültü giderici sistem çıkışı, adaptif süreç için hata işareti olarak yerine getirilmektedir[1].

n_0 gürültüsünün veya s işaretinin daha önceki bazı bilgileri gözönüne alınarak n_1'e

süzgeç tasarlanmadan önce veya gürültü giderme işareti y'yi üretmeye adapte olmadan önce gerek duyulmaktadır. İster istatistikî ister deterministik olsun s, n_0 veya n_1 bağıntıları daha önce bilinen bir bilgiye ihtiyaç duymamaktadır.

s, n_0, n_1 ve y'nin istatistik olarak sabit olduğu ve sıfır anlamına geldiği, s'nin n_0 ve n_1 ile ilişkisiz olduğu ve n_1'in n_0 ile ilişkili olduğu kabul edilirse ve Eşitlik (2.10)'u Eşitlik (2.15)'de yerine koyarsak Eşitlik (2.16) elde edilmektedir.

$$\varepsilon(n) = s(n) + n_0(n) - y(n) \tag{2.16}$$

Eşitlik (2.16)'nın karesi alınarak aşağıdaki ifade elde edilir.

$$\varepsilon^2 = s^2 + (n_0 - y) + 2s(n_0 - y) \tag{2.17}$$

Eşitlik (2.17)'de her iki tarafında beklenen değerleri alınıp ve s'nin n_0 ve y ile ilişkili olmadığı fark edildiğine göre;

$$\begin{aligned} E[\varepsilon^2] &= E[s^2] + E[(n_0 - y)^2] + 2E[s(n_0 - y)] \\ &= E[s^2] + E[(n_0 - y)^2] \end{aligned} \tag{2.18}$$

yazılabilir. $E[\varepsilon^2(n)]$'nin minimizasyonu, y(n) adaptif süzgecin çıkışı ve ana girişe verilen $n_1(n)$ gürültü işareti arasındaki farkı minimize eden eşitliktir. Çıkış gücü,

$$E_{\min}[\varepsilon^2] = E[s^2] + E_{\min}[(n_0 - y)^2] \tag{2.19}$$

süzgeç minimize edilmiş $E[\varepsilon^2]$'ye ayarlandığında $E[(n_0\text{-}y)^2]$'de minimize edilmiş olur. Daha sonra süzgeç çıkışı y, n_0 ana gürültüsünün, en iyi en küçük kareler tahmini olur. Bundan başka Eşitlik (2.16)'dan bu yana $E[(n_0\text{-}y)^2]$ minimize olduğunda, $E[(\varepsilon\text{-}s)^2]$'de minimize olmaktadır ve Eşitlik (2.20) elde edilmektedir.

$$(\varepsilon - s) = (n_0 - y) \tag{2.20}$$

Adaptif süzgeçlerin verilen ayarı, yapısı ve referans girişi için, bütün çıkış gücünü minimize etmek üzere süzgeci ayarlamak ve adapte etmek, s işaretinin tahmini en iyi en küçük karesinin ε çıkışına eşdeğer olmasını sağlar. ε çıkışı genellikle s işaretiyle toplanan bazı gürültüleri içerecektir. Eşitlik (2.16)'dan $(n_0\text{-}y)$ tarafından çıkış gürültüsü verilir. Minimize edilmiş $E[\varepsilon^2]$, $E[(n_0\text{-}y)^2]$ minimize ettiği için, minimize edilmiş tüm çıkış gücü, çıkış gürültü gücünü minimize eder ve çıkıştaki işaret sabit kalır. Minimize edilmiş bütün çıkış gücü çıkış işaret gürültü oranını maksimize eder.

Eşitlik (2.18)'de en küçük olası çıkış gücünün $E_{\min}[\varepsilon^2] = E[s^2]$ olduğu görülmektedir.

Bu başarıldığı zaman $E[(n_0-y)^2]=0$ dır. Bundan dolayı, $y=n_0$ $\varepsilon=s$ dir. Bu durumda, minimize edilmiş çıkış gücü, çıkış işaretinin gürültüden daima ayrılmasına sebep olur.

Diğer taraftan referans giriş ana girişten tamamen ilişkisiz olduğu zaman süzgeç kendi kendini kapatacak ve çıkış gürültüsü artmayacaktır. Bu durumda, süzgeç çıkışı y ana girişle ilişkisiz olacaktır. Çıkış gücü;

$$E[\varepsilon^2] = E[(s+n_0)] + 2E[-y(s+n_0)] + E[y^2]$$
$$= E[(s+n_0)] + E[y^2] \qquad (2.21)$$

şeklinde yazılabilir. Çıkış gücünün minimize edilmesi $E[y^2]$ nin minimize edilmesini gerektirir ki bu $E[y^2]$ yi sıfıra getirerek, tüm ağırlıklar sıfır yapılarak başarılır.

$E[\varepsilon^2(n)]$'nin minimizasyonu adaptif süzgeç katsayılarının güncelleştirilmesiyle başarılabilir. En sık LMS ve NLMS algoritmaları kullanılmaktadır. Sonraki sahip olunan avantaj adım büyüklüğünün nispeten giriş işaretinin genliğinden farklı olmasıdır. Şekil 2.9'daki tasarıma göre LMS algoritması Eşitlik (2.22) ve NLMS algoritması da Eşitlik (2.23)'de verildiği gibi tanımlanmaktadır [47].

$$w_k(n+1) = w_k(n) + \mu e(n) x_r(n) \qquad (2.22)$$

$$w_k(n+1) = w_k(n) + \frac{\mu e(n) x_r(n)}{\sum_{k=0}^{N} x_r{}^2(n-k)} \qquad 0 \le k \le N \qquad (2.23)$$

Burada μ, adım büyüklüğüdür.

2.3. LMS Algoritması

LMS algoritması, 1959'da Widrow ve Hoff tarafından desen tanıma makinalarında kullanılmıştır [2]. LMS algoritması, stokastik türeve dayalı bir algoritmadır. Algoritmanın en önemli özelliği çok basit bir yapıya sahip olmasıdır. Buna ilaveten, korelasyon fonksiyon hesaplamalarını ve matris terslendirmelerini içermezler [2].

2.3.1. LMS Algoritmasının Kararlılığı

LMS algoritmasında geribesleme olduğundan bu durum algoritmanın kararsızlığına sebep olabilmektedir. Bu durumu önlemek için adım büyüklüğü aşağıda verildiği gibi

alınabilir.

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{2.36}$$

Burada λ_{max} , R korelasyon matrisinin değeridir ve değeri aşağıda tanımlandığı gibi alınabilir.

$$\lambda_{\max} < tr[R] \tag{2.37}$$

2.3.2. LMS Algoritmasının Yapısı

LMS algoritması doğrusal bir adaptif süzgeçleme algoritması olup iki temel süreci ihtiva etmektedir:

1.) Süzgeçleme süreci, ***i.)*** verilen bir giriş işaretine [x(n)] karşın elde edilecek olan süzgeç çıkışının [y(n)] hesaplanması ve ***ii.)*** bu çıkış ile arzu edilen cevabın karşılaştırılarak bu iki işaretin farkını temsil eden bir kestirim hatasının [e (n)] oluşturulması işlemlerini ihtiva etmektedir.

2.) Adaptif süreç, kestirim hatasına göre, süzgeç parametrelerinin hata fonksiyonunu minimize edecek şekilde otomatik olarak ayarlanmasını içermektedir.

Bu iki sürecin bir araya gelmesiyle Şekil 2.10'da gösterildiği gibi bir geri besleme döngüsü oluşacaktır.

Şekil 2.10'daki yapı incelendiğinde, dört farklı işaretin tanımlandığı görülmektedir:

1.) x(n) : birbiri ile bağlantılı geciktirilmiş girişler vektörü,

$$x(n) = [x(n)x(n-1)x(n-2)...x(n-N+1)]^T \tag{2.24}$$

burada T matris transpozunu, x(n) ise Nx1 boyutunda bir vektörü ve N enine süzgeç yapısında kullanılan geciktirici sayısını yani süzgecin derecesini temsil etmektedir.

2.) d (n) : *n* anındaki arzu edilen işaret

3.) y (n) : *n* anındaki süzgeç çıkışı,

$$y(n) = w^H(n)x(n) \tag{2.25}$$

burada w(n), Nx1 boyutlu ağırlıklar vektörü veya katsayı vektörü olarak adlandırılır ve aşağıdaki gibi verilmektedir:

$$w(n) = [w_0(n)w_1(n)w_2(n)...w_{N-1}(n)]^T \tag{2.26}$$

burada H, Hermitian transpozunu belirtmektedir.

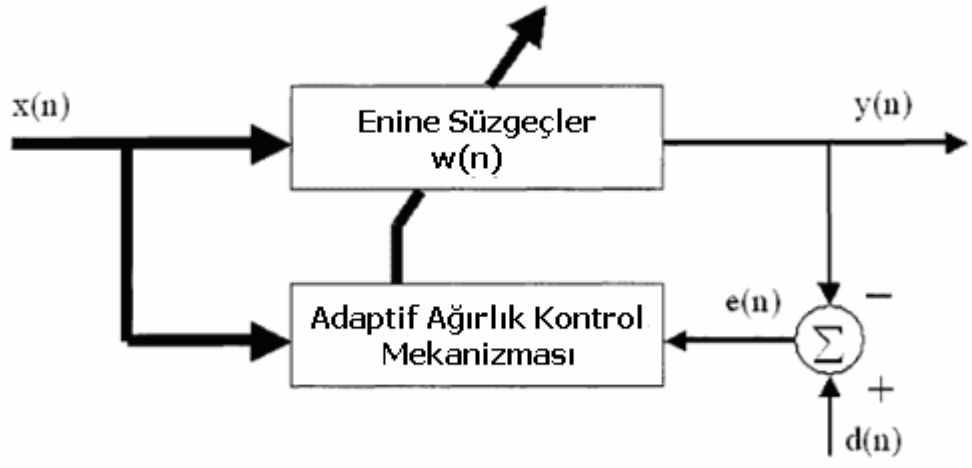
4.) e(n) : *n* anındaki kestirim hatası,

$$e(n) = d(n) - y(n) \tag{2.27}$$

Eşitlik (2.25), Eşitlik(2.27)'de yerine konulursa:

$$e(n) = d(n) - w^H(n)y(n) \tag{2.28}$$

elde edilir.



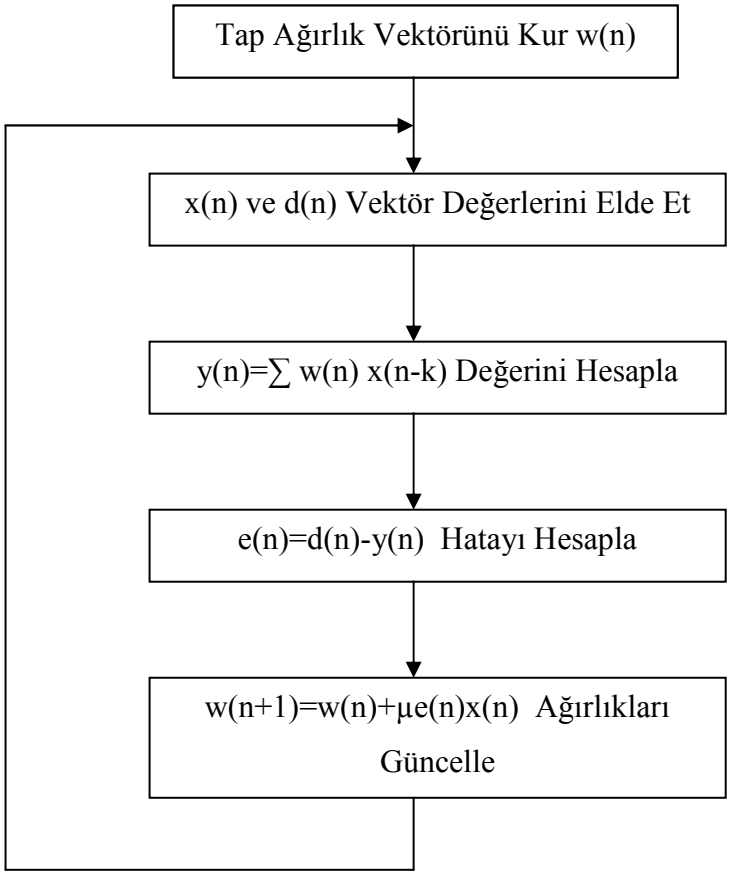
Şekil 2.10. Adaptif enine süzgecin blok diyagramı.

LMS algoritmasının temelini, karesel hatanın beklenen değerinin minimize edilmesini temel alan eski bir optimizasyon algoritması olan en hızlı iniş (steepest-descent-en dik çıkış) algoritması oluşturmaktadır. Bu nedenle, amaç fonksiyonunu temsil eden ortalama karesel hata (MSE) aşağıdaki gibi tanımlanmıştır:

$$J(n) = \mathrm{E}[e(n)e^*(n)] = \mathrm{E}[|e(n)|^2] \tag{2.29}$$

burada E[ ] beklenen değerdir. En hızlı iniş algoritmasının bir diğer özelliği ise, w (n)

ağırlık değerleri türev vektörünün (ortalama karesel hatanın türevi) tam zıt doğrultusunda hesaplanırlar ve bu hesaplama sonucunda da genelde optimum çözümler elde edilir.



Şekil 2.11. LMS algoritmasının akış diyagramı.

En hızlı iniş metoduna göre, (n+1) anındaki ağırlık değerleri (katsayılar) basit bir geribesleme yöntemi ile güncellenirler,

$$w(n+1) = w(n) + \mu[-\nabla J(n)] \tag{2.30}$$

burada μ, adım büyüklüğü olarak adlandırılan pozitif bir sabittir ve ∇J(n) ise n anındaki amaç fonksiyonunun türevidir. ∇J(n), türev vektörü Eşitlik (2.31) ile verilmektedir:

$$\nabla J(n) = -2P - 2R(n)w(n) \tag{2.31}$$

burada P, x(n) girişi ile d(n) istenen cevap arasındaki çapraz ilişki vektörüdür ve R, x(n) giriş vektörünün korelasyon matrisidir. P ve R değerleri, aşağıda verilen Eşitlik (2.32)

ve Eşitlik (2.33) kullanılarak hesaplanmaktadır:

$$\hat{P} = x(n)d^*(n) \tag{2.32}$$

$$\hat{R} = x(n)x^H(n) \tag{2.33}$$

(2.31), (2.32) ve (2.33) eşitlikleri, Eşitlik (2.30)'da kullanılırsa:

$$w(n+1) = w(n) + 2\mu x(n)[d^*(n) - x^H(n)w(n)] \tag{2.34}$$

elde edilir. Eşitlik (2.28) ve Eşitlik (2.34) birleştirilirse:

$$w(n+1) = w(n) + \mu e^*(n)x(n) \tag{2.35}$$

elde edilir. Eşitlik (2.35) LMS algoritmasının en basit halini göstermektedir. Şekil 2.11'de LMS algoritmasının akış diyagramı gösterilmektedir.

2.3.3. LMS Algoritmasının Öğrenme Eğrisi

LMS algoritmasının başarım ölçümlerinden biri aşağıda Eşitlik (2.38) ile verilen tahmini çıkış hatası yani öğrenme eğrisidir.

$$\text{Tahmini Hata Çıkışı} = 20\log_{10}(|e(n)|) \tag{2.38}$$

2.3.4. LMS Algoritmasının Dezavantajları

LMS algoritmasının en büyük iki dezavantajı;

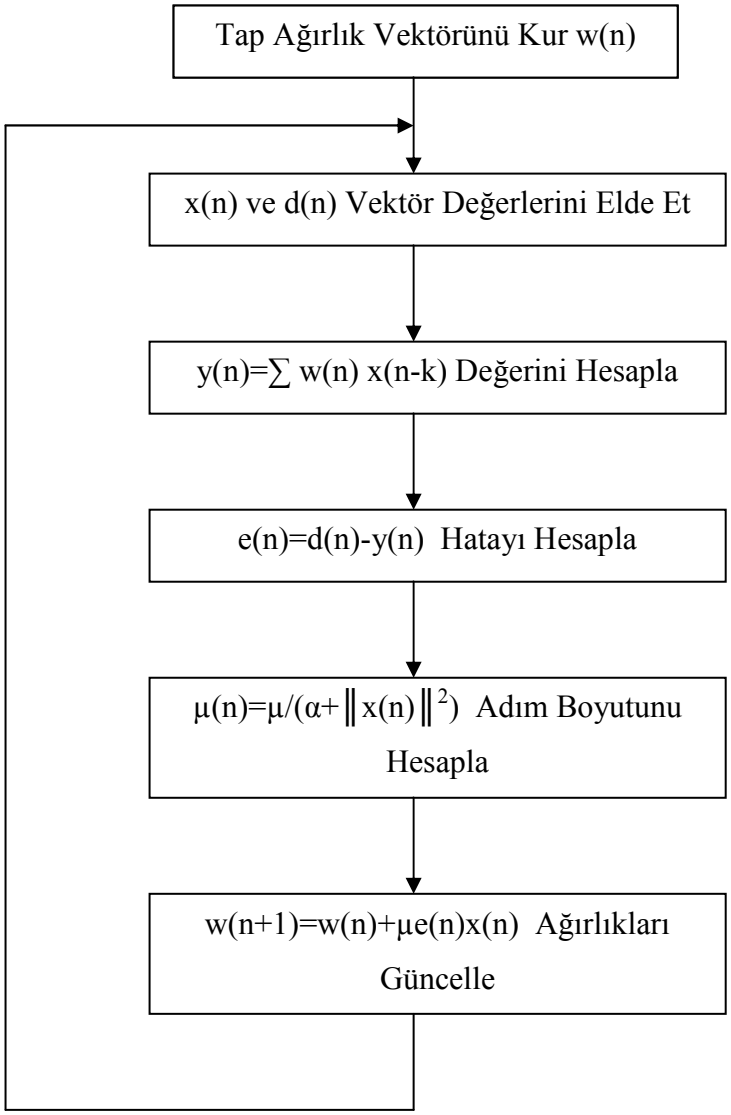
- Yakınsama hızı, μ adım büyüklüğüne bağlıdır.
- Korelasyon matrisi R'nin özdeğer yayılmasına olan hassaslığı (ki bu küçük özdeğerin büyük özdeğere oranıdır).

2.4. Normalize Edilmiş LMS Algoritması (NLMS)

LMS algoritmasında (n+1). iterasyonda uygulanan katsayı vektörünün bir önceki iterasyona göre iyileştirmesi işlemi 3 terimin çarpımından oluşmaktadır. Bunlar:

μ : adım büyüklüğü parametresi, e (n) : kestirim hatası, x (n) : giriş vektörüdür. Görüldüğü gibi iyileştirme giriş vektörü ile doğru orantılıdır. Bu nedenle giriş vektörünün boyutu büyük olduğunda algoritmanın hata başarımı düşer. Bunun üstesinden gelebilmek için, bütün iterasyonlarda sabit değerli bir adım büyüklüğü kullanan LMS algoritması yerine tüm iterasyonlarda zamanla değişen adım büyüklüğünü temel alan Normalize edilmiş LMS (NLMS) algoritması kullanılır.

Gerçekte, (n+1). iterasyondaki katsayı vektörüne uygulanan iyileştirme, n. iterasyondaki giriş vektörünün karesel Öklid normu ile gerçekleştirilir. NLMS algoritmasında adım büyüklüğü Eşitlik (2.39) ile verilen şekle dönüşmüş olur:



Şekil 2.12. NLMS algoritmasının akış diyagramı.

$$\mu(n) = \frac{\mu}{\alpha + \|x(n)\|^2} \tag{2.39}$$

Burada α, giriş vektörü x(n) çok küçük değerli olduğunda oluşacak nümerik işlem karmaşasını ortadan kaldırmak için kullanılan küçük değerli bir sayıdır ve

$\| x(n) \|^2 = \sum_{i=1}^{N} x_i^2(n)$ ise giriş vektörünün karesel Öklid normudur.

LMS algoritmasının düzenlenmiş versiyonuna Normalize edilmiş LMS Algoritması denilmektedir ve NLMS'nin akış diyagramı da LMS algoritmasınınkine çok benzemektedir. Şekil 2.12'de NLMS algoritmasının akış diyagramı gösterilmektedir.

2.5. SNR (Signal-To-Noise Ratio – İşaret Gürültü Oranı)

İşaret gürültü oranı (SNR), desibel olarak işaret ile gürültü arasındaki gürültünün derecesinin ölçümüdür.

$$SNR = 10\log_{10}(P_S / P_N)(dB) \tag{2.40}$$

Eşitlik (2.40)'daki gibi ifade edilir ve P_S ve P_N, sırasıyla işaretin ve gürültünün gücüdür ve sırasıyla Eşitlik (2.41) ve Eşitlik (2.42)'deki gibi tanımlanır.

$$P_S = \sum_{n=1}^{N} s^2(n) \tag{2.41}$$

$$P_N = \sum_{n=1}^{N} n_1{}^2(n) \tag{2.42}$$

ki burada Şekil 2.9'daki gibi; s(n) gürültüsüz (noise-free) işaret, x(n) referans işarettir. N verilen ölçüm periyotlu s(n) ve $n_1(n)$ de örneklerin toplam sayısıdır.

3. BÖLÜM

FARKSAL GELİŞİM ALGORİTMASI

Farksal gelişim algoritması evrime dayalı bir algoritmadır. Evrim prensibine dayalı teknikler bilinen optimizasyon metotlarının noksanlıklarının üstesinden gelmek için kullanılmaktadır [33,48–51]. Özellikle geçen 20–30 yılda evrim prensibine dayalı algoritmalara ilgi oldukça artmış ve bu konularla ilgili araştırmacıların ve literatürdeki çalışmaların sayısı hızla çoğalmıştır. Bu algoritma ilk kez K.Price tarafından 1995'de tanıtılmıştır. 1996 Mayıs'ında Nagoya'da yapılan 1stICEO (First International Contest of Evolutionary Computation-Evrimsel Hesaplamanın Birinci Uluslar arası Yarışması) konferansında, FG gerçek değerli test fonksiyonlarının çözümü kullanılmıştır. Son zamanlarda bu prensibe dayalı tekniklerin hepsini temsilen ortak bir terim olarak evrimsel hesaplama terimi yaygın olarak kullanılmaya başlanmıştır. Bu sınıfa giren algoritmalara örnek olarak genetik algoritmalar, evrimsel programlama (evolutionary programming), evrimsel stratejiler (evolution strategies) ve FG algoritması vs. sayılabilir [53–56].

3.1. Evrimsel Algoritmalar

FG, daha hızlı optimizasyon için Genetik Algoritmanın (GA) geliştirilmiş versiyonudur. FG değişen nokta sayılarının gerçek kodlarını kullanmaktadır. FG'nin avantajlarının içinde basit yapısı, kullanım kolaylığı, hızı ve güçlülüğü sayılabilir. Price&Storn, FG'nin on farklı stratejisini gerçekleştirmişlerdir. Temel olarak, FG'de iki popülasyon vektörü arasındaki ağırlık farkları, üçüncü vektöre eklenmektedir. Ayrıca literatürde bu paradigmaların çeşitli özelliklerini birlikte kullanan çok sayıda hibrid sistemler de mevcuttur. Bundan dolayı sınıflandırma yapmak oldukça zordur. Bir problemi çözmede kullanılacak herhangi bir evrimsel algoritma aşağıdaki beş elemana ihtiyaç duymaktadır:

- Problem için çözümlerin genetik temsili,
- Çözümlerin başlangıç popülasyonunu oluşturacak bir yöntem,
- Çözümleri uygunluk açısından değerlendirmeye tabii tutacak değerlendirme fonksiyonu yani çevre,
- Genetik kompozisyonu değiştirecek operatörler,
- Kontrol parametrelerinin değerleri (popülasyon büyüklüğü, operatörleri uygulama ihtimalleri vs.).

Evrimsel algoritmalarda tek bir bireyle (çözümle) değil, bireylerin popülasyonu P(t) ile ilgilenilmektedir. Her birey mevcut problem için muhtemel bir çözümü temsil etmekte ve bir veri yapısı (S) olarak tanımlanmaktadır. Her birey uygunluk (fitness) ölçüt değerini belirlemek amacıyla değerlendirilerek daha uygun bireyleri seçmek suretiyle yeni popülasyon (P(t+1)) oluşturulmaktadır (seçme adımı). Yeni popülasyonun bazı bireyleri, yeni çözümler oluşturmak için bazı operatörler vasıtasıyla transformasyona tabi tutulurlar (değiştirme adımı). Bir evrimsel algoritmanın temel adımları aşağıda verilmiştir.

Adım 1. P(t)'yi başlangıç durumuna ayarla.
Adım 2. P(t)'yi değerlendir.
Adım 3. Durdurma kriteri sağlanıncaya kadar aşağıdaki işlemleri gerçekleştir
  3.1. Önceki popülasyonda bulunan bireylere seçme işlemi uygulayarak yeni popülasyon oluştur.
  3.2. Yeni popülasyonu değiştir.
  3.3. Yeni popülasyonu değerlendir.

Evrimsel hesaplama teknikleri arasında oldukça benzerlikler olmasına rağmen birçok farklılıklar da mevcuttur. Örneğin, çözümlerin temsili için farklı veri yapıları kullanırlar ve dolayısıyla kullanılan genetik operatörler de farklı olabilir. Yine yukarıda verilen temel adımlarda 3.1 ve 3.2 adımları yer değiştirebilir. Örneğin, evrimsel stratejilerde ilk önce popülasyon değiştirilir ve sonra yeni popülasyon seçme işlemi ile oluşturulur. Aynı teknikte bile değişik alternatifler mevcuttur. Örneğin, FG algoritmasında bireylerin seçimi için kullanılabilecek çok sayıda seçme metodu bulunmaktadır.

FG çeşitli alanlarda başarılı şekilde uygulanır. FG algoritmasının çeşitli uygulama alanları; sayısal süzgeç tasarımı [17], fuel ethanol ürünün problemlerinin yapılan bulanık kararlılığı [57], bulanık lojik kontrolör tasarımı [58], grup fermantasyon işlemi [59,60], çoklu sensör birleşmesi [61], sürekli polimer reaktörün dinamik optimizasyonu [62], akan yatak reaktöründe ısı iletim parametrelerinin tahmini [63], ısı değişiminin optimal tasarımı [64,65], ısıyla bütünleşmiş damıtma sistemin optimizasyonu ve sentezi [66], doğrusal olmayan fonksiyonların optimizasyonu [67], termik kraker (CRACKER) operasyonun optimizasyonu [68] gibi sayılabilir.

3.2. Farksal Gelişim Algoritması

FG algoritması basit, ama güçlü popülasyon tabanlı bir algoritmadır [54]. Özellikle tamamen düzenlenmiş uzayda tanımlı ve gerçek değerli tasarım parametrelerini içeren fonksiyonları küresel olarak optimize etmek amacıyla kullanılan bir direkt araştırma algoritmasıdır. Gerçek parametreli optimizasyon, bilimde, mühendislikte ve iş ortamlarında karşılaşılan pratik problemlerin önemli ve geniş bir sınıfını oluşturmaktadır. Aerodinamik şekillerin optimizasyonunda, IIR süzgeç tasarımında, Sinir ağlarının öğrenmesi gibi farklı uygulamalarda dinçliği ve gücü gösterilmiştir [17,69,70]. Zor optimizasyon problemleriyle karşılaşıldığında genellikle ilk yapılması gereken, probleme özel sezgisel bir yaklaşım tekniği belirlemek olacaktır. Uzman bilgisinin tasarıma ilave edilmesine olanak sağlayan böyle teknikler, gelişime dayalı algoritmalar gibi genel metotlarla karşılaştırıldığında oldukça üstün başarım göstermektedirler. Örneğin, amaç fonksiyonu doğrusal polinom formunda ise bu tür problemi çözmek amacıyla geliştirilmiş Simpleks metodu, Karmakar algoritması gibi başarılı yöntemler mevcuttur. Bu yöntemler, bu tip problemlerde yüzlerce hatta binlerce tasarım değişkeninin bulunması durumunda bile küresel optimal çözümleri bulabilme kabiliyetine sahiptir. Amaç fonksiyonu doğrusal bir polinom şeklinde değilse ve probleme özel metotlar yoksa o zaman genellikle Levenberg-Marquardt algoritması, Branch and Bound metodu gibi yöntemlere başvurulmaktadır.

Maalesef, doğrusal olmama, gürültü, yüksek boyutluluk, çok modluluk, sınırlamalar, değişmeme (düz olma) ve türev alınamama gibi özellikler tek başlarına veya birlikte bu tür deterministik metotların başarımlarını çok olumsuz etkilemektedirler. Stokastik algoritmalar gibi gelişime dayalı algoritmalar, sınırlamaların çoğunun üstesinden

gelebilmektedirler. Ancak bu tür genel metotlar, probleme özel bilgiyi kullanarak tasarımlarına aktaramazlarsa bunların da başarımı nispeten düşmektedir. Optimizasyon işleminin tamamen düzenlenmiş uzay domeninde sınırlandırılma bilgisi, kendi başına gelişime dayalı algoritmaların başarımını artırmak için kullanılabilecek probleme özel önemli bir bilgidir. Tipik bir gelişime dayalı algoritmanın temel basamakları aşağıdaki gibidir:

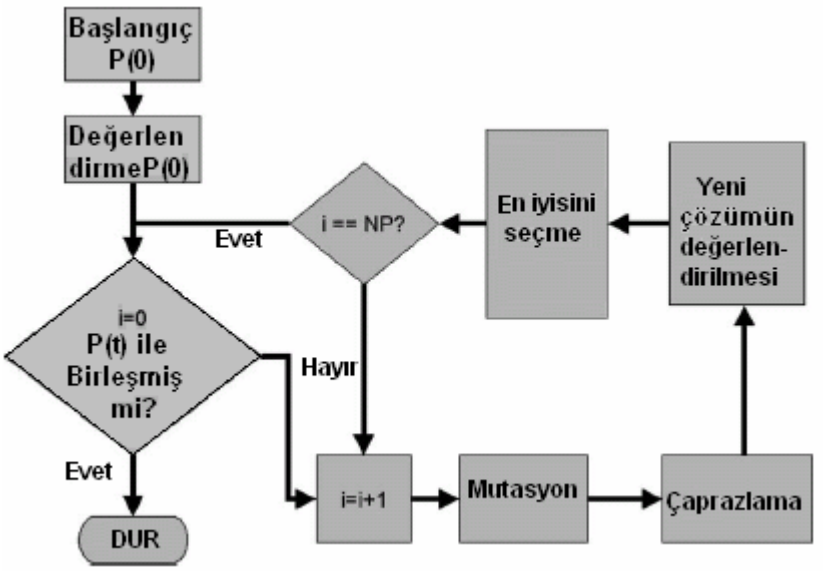
Adım 1. Başlangıç popülasyonunun oluşturulması
Adım 2. Değerlendirme
Adım 3. TEKRARLA
Adım 3.1. Seçme
Adım 3.2. Mutasyon ve Yeniden birleştirme
Adım 3.3. Değerlendirme
DURDURMA KRİTERİ SAĞLANINCAYA KADAR

Gelişime dayalı algoritmaların genel amaçlı sayısal optimizasyon algoritmaları olarak incelenmesi ve değerlendirilmesi ile ilgili çalışmalara literatürde oldukça sık rastlanmaktadır. Ancak bu tartışmaların çoğu, orijinalinde bu amaç için tasarlanmamış tümleşik (combinatorial) tip algoritmalarla elde edilen sonuçlar üzerinedir. FG algoritması ise bir ayrık optimizasyon algoritması değil özellikle sayısal optimizasyon için geliştirilmiş bir gelişim algoritmasıdır. Bu algoritma yeni, ama sadece basit olmayan aynı zamanda oldukça da etkili olan bir mutasyon işlemi uygulamaktadır. Daha önce tanımlanmış, olasılık dağılım fonksiyonuna dayalı olarak çalışan genetik algoritma gibi gelişim tabanlı algoritmaların tersine FG algoritması rastgele olarak seçilmiş amaç vektör çiftlerinin farklarına dayalı bir mutasyon işlemi kullanır. FG algoritmasında kullanılan basit mutasyon işlemi, algoritmanın başarımını geliştirmekte ve onu daha güçlü yapmaktadır. Bu özelliğinin yanı sıra diğer özellikleri için de şunlar söylenebilir: Hızlı, basit, kolayca kullanılabilir ve değiştirilebilir, etkili küresel optimizasyon kabiliyetli, doğal olarak paralel, kayan nokta formatına bağlı hassasiyet sınırlamalı, matris çarpımları ve sıralama işlemleri olmadığı için hesaplama maliyeti açısından avantajlı, daha önceden tanımlanmış herhangi bir olasılık dağılımlı mutasyon kullanmamaktadır. Tamsayı, ayrık ve karışık parametre optimizasyonuna kolaylıkla uyarlanabilir, amaç fonksiyonunun veya sınırlama fonksiyonlarının türevine gerek duymaz, düz yüzeylerde çalışabilir, gürültülü ve zamana bağlı amaç fonksiyonları için

kullanılabilir, tek bir koşmada alternatif çözümler üretebilir ve özellikle doğrusal olmayan sınırlamalı optimizasyon problemlerinde etkilidir.

3.3. Algoritmanın Temel Adımları

FG algoritmasının temel adımları açık bir şekilde aşağıda verilmektedir. Çoğu açıdan bu algoritma tipik bir gelişim algoritmasıdır. Adım 2'de görüleceği üzere algoritma başlangıçta, NP (Number of Population-Popülasyon büyüklüğü) tane D (Dimension-boyut) boyutlu amaç değişken vektörlerinin x_{ijG} rastgele dağıtılmış başlangıç popülasyonunu (P_0) üretir. Randj [0,1] terimi, 1 ile 0 arasında uniform şekilde dağılmış rastgele değişkeni temsil etmektedir. *j* indisi *j*'nin her bir değeri için yeni rastgele sayı üretildiğini göstermektedir. *i* indisi ise rastgele değerin her bir değeri için yeni bir rastgele sayı üretildiğini göstermektedir. h_i ve l_o indisleri üst ve alt başlangıç parametre sınırlarını temsil etmektedir. Oluşturulan bu popülasyon G_{max} jenerasyonu(Maximum Generation-Maksimum Jenerasyon, İterasyon) boyunca mutasyon, çaprazlama ve seçme operatörleri uygulanarak başarılı bir şekilde geliştirilir [71,72]. FG algoritmasının en önemli özelliği deneme vektörünün üretilme şeklidir [73,74]. Şekil 3.1'de Farksal gelişim algoritmasının akış diyagramı verilmektedir.



Şekil 3-1. Farksal gelişim algoritmasının akış diyagramı.

Algoritmanın çalışması aşağıda anlatılmaktadır:

Adım 1. Kontrol parametrelerinin

D, G_{max}, NP≥4, F∈(0,1+), CR∈ [0,1], değerlerini ve parametre sınırlarını $\bar{x}^{(lo)}, \bar{x}^{(hi)}$ ata.

Adım 2. Başlangıç popülasyonunu oluştur

$$\forall i \le NP \wedge \forall J \le D : x_{j,i,G=0} = x_j^{(lo)} + rand_j[0,1].(x_j^{(hi)} - x_j^{(lo)})$$
$$i = (1,2,...,NP), j = (1,2,...,D), G = 0, rand_j[0,1] \in [0,1]$$

Belirli bir durdurma kriteri sağlanıncaya kadar aşağıdaki adımları tekrarla

Adım 3. Mutasyon ve yeniden birleştirme

$r_1, r_2, r_3 \in \{1,2,...,NP\}, r_1 \neq r_2 \neq r_3 \neq i$(rasgele seçilmiş)

$j_{rand} \in \{1,2,...,D\}$, (rastgele seçilmiş)

$$\forall j \le D, u_{j,i,G+1} = \begin{cases} x_{j,r_3,G} + F.(x_{j,r_1,G} - x_{j,r_2,G}) \\ \quad \text{eğer } (rand_j[0,1) < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{diğer durumlarda} \end{cases}$$

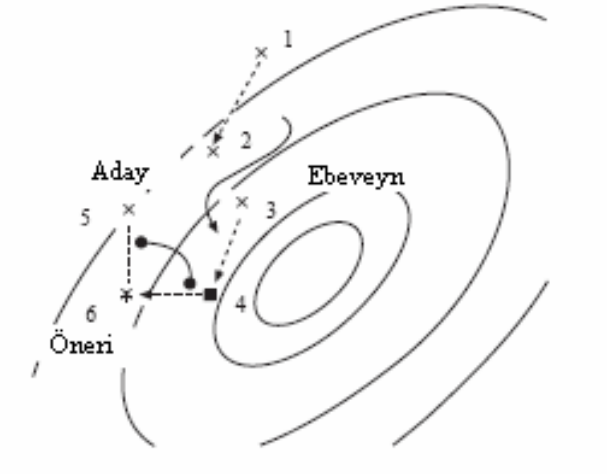
Adım 4. Seçme

$$\bar{x}_{i,G+1} = \begin{cases} \bar{u}_{i,G+1} & \text{eğer } f(\bar{u}_{i,G+1}) \le f(\bar{x}_{i,G}) \\ \bar{x}_{i,G} & \text{diğer durumlarda} \end{cases}$$

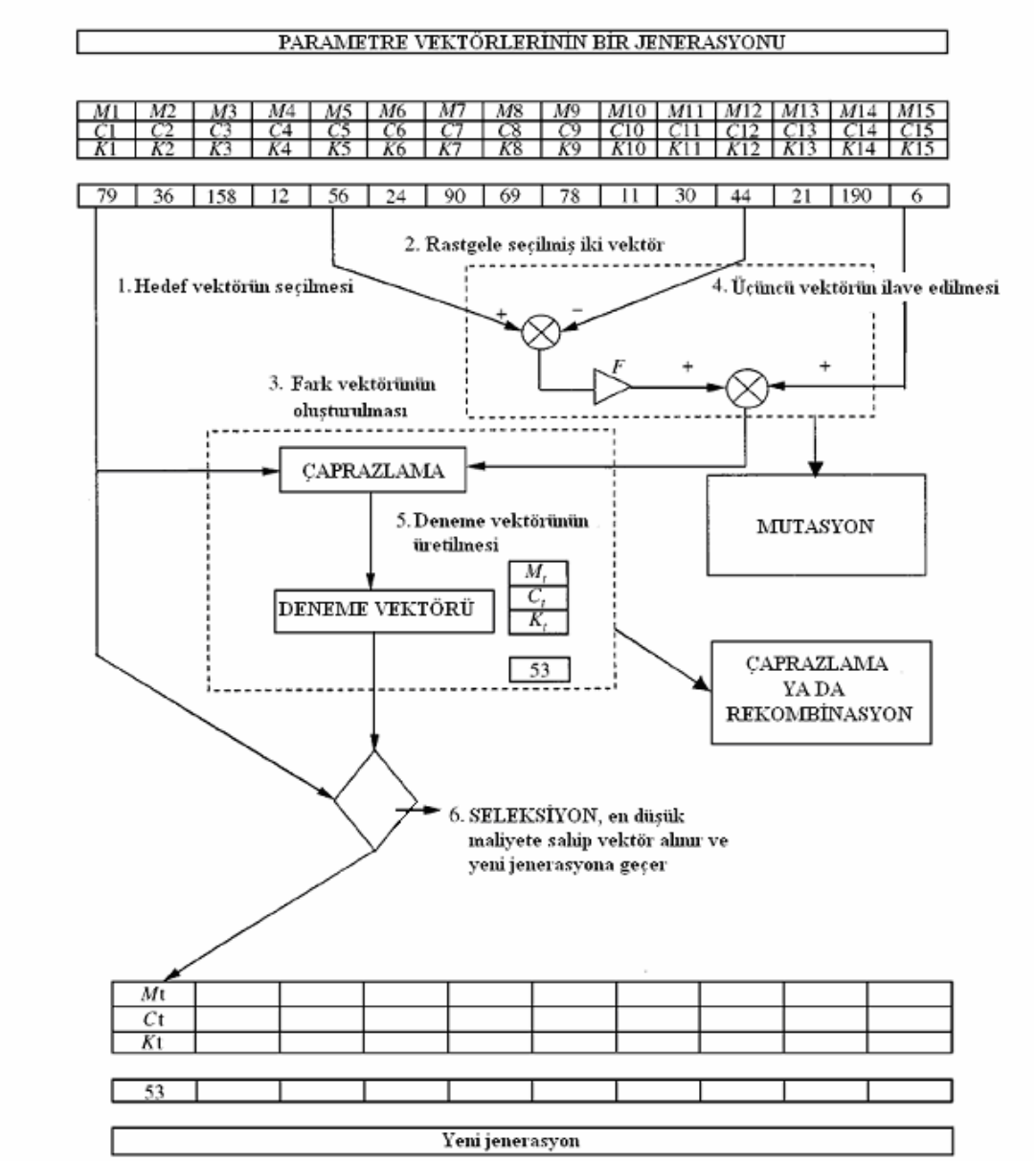
FG algoritması, bir ebeveyn (amaç) vektöründen x_{ijG}, bir deneme vektörü $u_{j,i,G+1}$ üretmek için mutasyon ve yeniden birleştirme (recombination) işlemlerinin her ikisini birlikte kullanmaktadır. r_1, r_2, r_3 indisleri, i indisinden farklı olan ve aynı zamanda birbirlerinden farklı rastgele seçilmiş popülasyon indisleridir. Burada i mevcut amaç vektörünü temsil etmektedir. Dolayısıyla adım 1'de belirtildiği gibi popülasyon büyüklüğü NP her zaman 3'ten büyük olmalıdır. Hem CR (Crossover Rate-Çaprazlama Oranı) hem de F değerleri kullanıcı tarafından belirlenecek olan kontrol parametreleridir. CR olasılığı temsil ettiği için 0 ile 1 arasında değerler almaktadır. F (Fitness Scaling-Uygunluk Ölçekleme) ise bir ölçekleme faktörüdür. $Rand_j$ [0,1] CR'den küçükse veya $j=j_{rand}$ olduğunda yeni (child-çocuk) parametre, rastgele seçilmiş üç tane vektörün doğrusal kombinasyonu olacaktır. Değilse, yeni parametre doğrudan kendi ebeveyninden alınacaktır. ,$j=j_{rand}$ şartı, yeni vektörlerin en az bir parametrede de olsa ebeveyn vektörlerin farklı olmasını garanti etmek amacıyla kullanılmaktadır.

Adım 4'te görüldüğü gibi her yeni vektör, amaç fonksiyonu vasıtasıyla değerlendirildikten sonra maliyeti ebeveyninin maliyeti ile karşılaştırılır. Şayet yeni vektör eşit veya daha düşük bir maliyete sahip olursa yeni vektör popülasyondaki ebeveyn vektörü ile yer değiştirir. Yoksa ebeveyn vektör yeni vektörle yer değiştirmeyip popülasyonda muhafaza edilir. Bir sonraki jenerasyonun üyeleri bu seçme işlemiyle seçildikten sonra gelişim çevrimi ya problem çözülene kadar ya da tüm vektörler bir noktaya yakınsayana kadar tekrar etmektedir.

Yeni bir çözümün üretilmesi Şekil 3.2'de gösterilmiştir. Şekil 3.2'ye göre; iki popülasyon üyesi (1,2) arasındaki fark üçüncü popülasyon üyesine (3) ilave edilmekte ve sonuç (4), aday üyeyle (5) çaprazlama işlemine tutulmaktadır. Böylece yeni çözüm (6) elde edilmektedir. Şayet yeni çözüm, adaydan daha iyiyse adayla yer değiştirmektedir. Algoritmanın şematik diyagramı Şekil 3.3'de verilmektedir.



Şekil 3.2. FG algoritmasında yeni bir çözümün elde edilmesi.



Şekil 3.3. FG algoritmasının şematik diyagramı.

3.3.1. Kodlama

Sayısal optimizasyonu amaçlayan çoğu gelişim algoritmaları sayısal parametreleri kodlamak amacıyla ikili tamsayıları kullanmaktadır. Ancak sayısal optimizasyon işlemlerinde tamsayı formatla parametre değerlerinin geniş dinamik sahasını verimli olarak temsil etmek pek mümkün olmamaktadır. Bunun için Gray kodlama gibi değişik kodlama türleri kullanılsa bile, hala bu tür yaklaşımlar geniş dinamik sahayı tanımlama kabiliyetinden yoksun kalmaktadır. Bu yüzden amaç vektörlerini kayan-noktalı sayılar kullanarak kodlama tercih edilen bir yöntem olmaktadır.

Bundan dolayı FG algoritması, gerçek parametreleri bilinen kayan-noktalı sayılar kullanarak kodlamakta ve bu parametreler arasındaki işlemleri, standart kayan-noktalı aritmetik mantığına göre gerçekleştirmektedir.

3.3.2. Popülasyon Yapısı ve Parametre Sınırları

FG algoritması, sabit büyüklükte popülasyon kullanmaktadır. Popülasyon, NPxD kayan-noktalı dizi formunda tanımlanmaktadır. Burada NP, popülasyondaki amaç vektörlerinin sayısına ve D de bir amaç vektöründeki parametrelerin sayısına karşılık gelmektedir. Gelişim algoritmasını paralel formda gerçekleştirirken, amaç vektörlerinin iki dizisi hafızada tutulur. Birinci dizi mevcut ebeveyn popülasyonu için diğeri ise yeni vektörleri ihtiva eden gelecek jenerasyonu oluşturmak amacıyla kullanılmaktadır. Başlangıç amaç vektörleri iyi tanımlanmış sınırlamalara sahip bir araştırma uzayından uniform dağılımla rastgele elemanlar seçilerek oluşturulur. Pratikte, başlangıç parametre sınırları genellikle fiziksel şartlara göre belirlenir. Fiziksel şartlar belirleyici değilse o zaman başlangıç alt ve üst parametre sınırları, küresel optimayı ihtiva eden bölgeleri kapsayacak kadar büyük seçilmelidir. Alt ve üst parametre sınırları başlangıç parametreleri için kullanıldıktan sonra algoritmanın başlangıçta verilen sınırlar ötesinde araştırma yapmasına imkân vermek amacıyla ihmal edilebilir. Bununla birlikte sınırlamalar, özellikle düzgün ya da düzgüne yakın ani değişim göstermeyen yüzeyler ile çalışıldığında sonsuza gidişin engellenmesi için gereklidir. Başlangıçta eğer iyi bir çözüm tespit edilebilmişse veya zaten mevcutsa, algoritma bu nominal çözüm etrafında dağılmış amaç vektörlerinden oluşan popülasyonla araştırmaya başlayabilir. Ancak bu şekildeki bir uygulama araştırmayı bölgeselleştireceğinden pek tavsiye edilmez.

Hatırlanması gereken önemli bir nokta, başlangıç parametre değerlerinin bir şekilde rastgele dağıtılmasıdır. Çünkü asıl araştırmayı yönlendiren, amaç vektörleri arasındaki farklardır. Başlangıç popülasyonu oluşturulurken uygun olmayan bir amaç vektörü kabul edilmeyip reddedilebilir ve tekrar üretilebilir. Yeni bir parametre değeri, katı bir sınırlamayı aşarsa o zaman parametreye müsaade edilen herhangi bir değeri atamak için çeşitli yöntemler mevcuttur. Bunların en basiti, parametreye değer olarak aşmış olduğu sınır değerini atamaktır.

3.3.3. Mutasyon

Parametre optimizasyonu açısından mutasyon işlemi, var olan bir amaç vektörünün bir veya daha fazla parametreleri üzerine, rastgele üretilmiş bir artışın ilave edilme işlemine karşılık gelmektedir. İyi bir mutasyon işleminin amacı, var olan amaç vektörlerini doğru zamanda doğru miktar ile doğru yönde hareket ettirecek artışları başka bir ifadeyle adımları üretmektir. Bu hedefi başarmak esas olarak adımları üretecek dağılımın karakteristiklerine bağlıdır.

FG algoritması, bir amaç vektörüne mutasyon işlemini, rastgele seçilmiş amaç vektörler çiftinin ağırlıklaştırılmış farkının bu amaç vektörüne ilave edilmesiyle gerçekleştirir. Algoritmanın basit mutasyon işlemi sık sık aritmetik çaprazlama işlemiyle karıştırılır.

Aritmetik çaprazlama, iki vektörün genel doğrusal kombinasyonu iken mutasyon üç vektörün doğrusal kombinasyonunun özel bir halidir. Her iki durumda da mevcut amaç vektörlerinin doğrusal kombinasyonu olan yeni vektörler üretilmektedir. Kısacası FG algoritması, gerçek parametre optimizasyonuna rastgele amaç vektörlerinin doğrusal kombinasyonlarını örnekleyen, mutasyon ve yeniden birleştirme gibi iki basit operasyon kullanan normal gelişime dayalı bir yaklaşımdır. Aritmetik çaprazlama ile mutasyon birbirlerine benzemelerine rağmen amaç vektörleri uzayını oldukça farklı şekilde araştırırlar.

3.3.4. Seçme

Seçme işlemi, yeni üretilen vektörlerin hangi şartlar altında popülasyona girebileceğini tanımlayan bir kriterdir. Örneğin, turnuva seçme işlemi rastgele seçilmiş vektör çiftleri arasında bir dizi yarışma gerçekleştirmek suretiyle yeni jenerasyonun üyesini belirlemektedir. Tipik olarak yarışan vektörler ebeveyn-aday popülasyonundan seçilir ve en fazla kazanan vektörlerin gelişmesine müsaade edilir.

Gelişim stratejisi, alternatif olarak birleştirilmiş ebeveyn-çocuk popülasyonunun en iyi başarım gösteren üyelerinin gelişimine müsaade edildiği deterministik bir seçme işlemi kullanmaktadır. Her iki seçme işleminin dezavantajı, her jenerasyon sonunda popülasyonun kalite değerlerine göre sıraya dizilme maliyetidir. FG algoritmasının seçme işleminde, yeni üretilen vektör ebeveynine göre daha gelişmiş veya en azından

aynı gelişme seviyesinde değilse ebeveyn vektör en az bir jenerasyon daha popülasyonda kalmaya devam etmekte ve başka bir vektörle yer değiştirmemektedir.

3.3.5. Yeniden Birleştirme

Mutasyon işlemi, temel olarak bir popülasyonu güçlü yapmaktan ve yeni araştırma bölgelerinin araştırılmasından sorumludur. Yeniden birleştirme veya çaprazlama işlemi ise tamamlayıcı bir işlemdir ve amacı, var olan amaç vektör parametrelerinden faydalanarak yeni vektörleri oluşturmak suretiyle araştırmanın başarılı olması için yardımcı olmaktır. Yeniden birleştirme etkisi üzerine oldukça ciddi çalışmalar ve tartışmalar mevcuttur. Ama kesin olan bir sonuç, uniform yeniden birleştirme işleminin araştırmanın hesaplama maliyetini önemli miktarda artırdığıdır. Ancak önemli miktarda da araştırmaya hızlılık sağlamaktadır.

FG algoritmasının ilk versiyonları özel uniform olmayan ayrık yeniden birleştirme işlemi kullanmıştır. Bu işlemde, ebeveynle değiştirilme ihtimali olan yeni çözüm oluşturulurken çözümün bazı elemanları ebeveynden bazıları ise mutasyona tabii tutulmuş vektörden alınmaktadır. Yeni aday çözümün oluşumunda, hangi vektörden daha çok elemanın alınacağını belirleyen frekansı kontrol etmek amacıyla hem üstel hem de binomial işlemler kullanılmıştır. Bu iki işlem, kullanıcı tanımlı, optimizasyon işlemi boyunca sabit tutulan ve çaprazlama oranı olarak adlandırılan kontrol parametresi aracılığıyla ayarlanır. Binomial uygulamada, 0 ile 1 arasında rastgele üretilen bir sayının CR'den küçük olması durumlarında elemanlar mutasyona uğramış çözümden, değilse ebeveynden alınmaktadır. Bunun tersine üstel uygulamada ise 0 ile 1 arasında rastgele üretilen bir sayının CR'den küçük olması anına kadar elemanlar ebeveynden, büyük olduğu andan itibaren de mutasyona uğramış çözümden alınmaktadır. Her iki durumda da yeni üretilen vektörün en az bir parametresinin ebeveynden farklı olması sağlanmaktadır [55].

3.4. Farklı FG Algoritmaları

Rastgele çözümler üretmek için popülasyondan fark ve bilginin yönünü çıkarmak son derece önemli yakınsama özelliklerine sahip adaptif bir şekil ile sonuçlanır. FG algoritması mutasyon ve çaprazlama operatörlerinin özelliklerine göre ikiye ayrılır. İlki, DE/rand/1/bin olarak gösterilen ve değişim vektörünün popülasyondan rastgele seçildiği

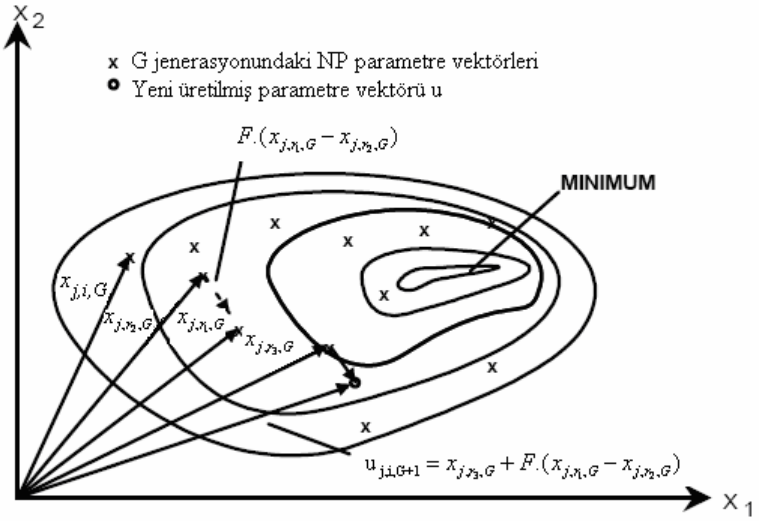
sadece fark vektörünün mutasyona tabi tutulduğu algoritmadır. İkinci FG algoritması ise kısaca DE/best/2/bin olarak gösterilen algoritmadır [50,72].

3.4.1. DE/rand/1/bin

FG algoritmasının ilk şekli olan DE/rand/1/bin'de her bir deneme vektörü $u_{i,j,G+1}$ Eşitlik (3.1)'deki gibi elde edilmektedir:

$$\begin{aligned} &r_1, r_2, r_3 \in \{1, 2, ..., NP\}, r_1 \neq r_2 \neq r_3 \neq i \text{(rasgele seçilmiş)} \\ &j_{rand} \in \{1, 2, ..., D\}, \text{(rastgele seçilmiş)} \\ &u_{j,i,G+1} = x_{j,r_3,G} + F.(x_{j,r_1,G} - x_{j,r_2,G}) \end{aligned} \tag{3.1}$$

F>0 olmak zorundadır. r_1,r_2,r_3 rasgele seçilmiştir, i'den farklı değerler almak zorundadır ve [1,NP] aralığındadır. F gerçel bir sayıdır ve $(x_{j,r_1,G} - x_{j,r_2,G})$ arasındaki farkı kontrol altına alan sabit bir faktördür. Şekil 3.4, DE/rand/1/bin'de rol oynayan fark vektörlerini gösteren iki boyutlu bir örneği ele almaktadır.



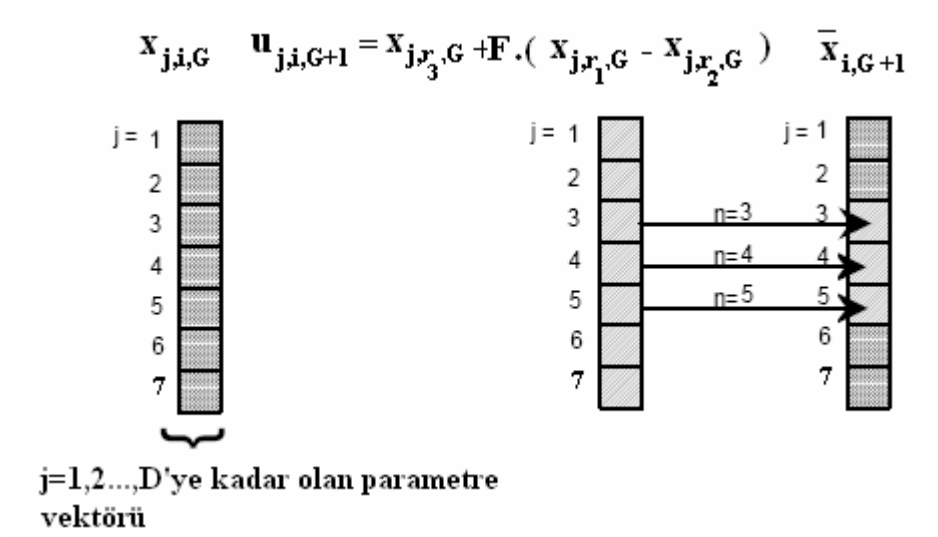
Şekil 3.4. DE/rand/1/bin'e göre $u_{j,i,G+1}$'i elde etmek için oluşan süreç ve dışhat(contour) çizgileriyle amaç fonksiyonunun gösterimi.

Parametre vektörleri arasındaki çeşitliliği artırmak için, vektör Eşitlik (3.2)'de verildiği gibi tanımlanmaktadır.

$$\bar{x}_{i,G+1} = \begin{cases} \bar{u}_{i,G+1} & \text{eğer j=}\langle n \rangle_D, \langle n+1 \rangle_D, ..., \langle n+L-1 \rangle_D \\ \bar{x}_{i,G} & \text{diğer durumlarda} \end{cases} \tag{3.2}$$

şeklinde tanımlanır. Burada $\langle\ \rangle_D$ işareti D modunda olduğunu göstermektedir.

Yani $\bar{x}_{i,G+1}$ vektör elemanlarının belirli bir dizisi $u_{j,i,G+1}$'e özdeştir, diğer elemanları ise $x_{j,i,G}$'nin orijinal değerlerini kazanır. Mutasyon işlemi gelişime dayalı algoritmalardaki çaprazlamayla aynıdır. Bu fikir Şekil 3.5'te D=7, n=3 ve L=3 için gösterilmektedir. n [1,D] aralığında rasgele seçilmiş bir tamsayıdır. L, [1,D] aralığında seçilmiş bir tamsayıdır. Bu rasgele sayılar n ve L yeni bir deneme vektörü elde etmek için kullanılmaktadır [50].



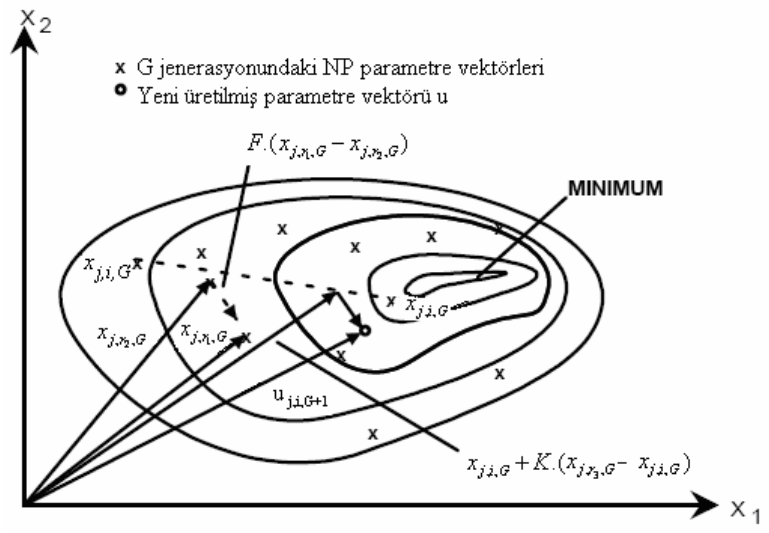
Şekil 3.5. D=7, n=3 ve L=3 için çaprazlama örneği.

3.4.2. DE/best/2/bin

Temel olarak, DE/best/2/bin DE/rand/1/bin ile aynı şekilde çalışmaktadır. Fakat $u_{j,i,G+1}$ Eşitlik (3.3)'deki gibi üretilir:

$$r_1, r_2, r_3 \in \{1,2,...,NP\}, r_1 \neq r_2 \neq r_3 \neq i \text{(rasgele seçilmiş)}$$
$$j_{rand} \in \{1,2,...,D\}, \text{(rastgele seçilmiş)}$$
$$\forall j \leq D, u_{j,i,G+1} = \begin{cases} x_{j,i,G} + K.(x_{j,r_3,G} - x_{j,i,G}) + F.(x_{j,r_1,G} - x_{j,r_2,G}) \\ \qquad \text{eğer } (\text{rand}_j[0,1) < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{diğer durumlarda} \end{cases} \tag{3.3}$$

Kontrol değişkeni K ile yöntem biraz değiştirilmiş olmaktadır. K kombinasyon faktörünü kullanmadaki ana fikir mevcut en iyi vektör $x_{j,i,G}$'yi de hesaba katarak DE/rand/1/bin'i geliştirmektir. Bu özellik kritik olmayan amaç fonksiyonlarında faydalı olabilmektedir. Şekil 3.6, yukarıdaki eşitlikle tanımlanan vektör-jenerasyon ilişkisini göstermektedir. Bütün sabitler DE/rand/1/bin'e özdeştir [50].



Şekil 3.6. DE/best/2/bin'e göre $u_{j,i,G+1}$'i elde etmek için oluşan süreç ve kontür çizgileriyle amaç fonksiyonunun gösterimi.

4. BÖLÜM

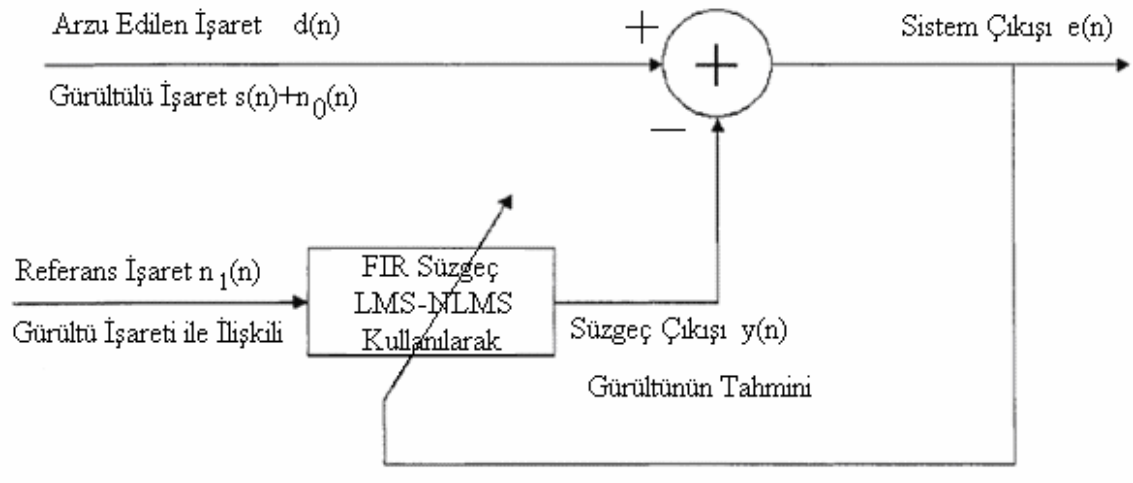
ADAPTİF FIR SÜZGEÇLERDE GÜRÜLTÜ GİDERME

Gürültü gidermek, birçok mühendislik dalı için önemli bir problemdir. Sayısal haberleşme sistemi uygulamalarında, gürültüden bilgi işaretini tamamen çıkarmak ve gürültüyü temizlemek önemli olmaktadır. Adaptif FIR süzgeçler, kararlılığı sebebiyle geniş bir alanda uygulama imkanı bulmuşlardır. Bu bölümde, gürültü gidermek için adaptif FIR süzgeç algoritmasının öncelikle klasik yapılardan LMS ve NLMS algoritmasıyla tasarlanması anlatılacak ve üç farklı işaret için Adaptif Gürültü Giderici (AGG)'nin elde ettiği sonuçlar verilecektir. Sonra gürültü giderme için bir optimizasyon algoritması olan FG algoritmasının, süzgeç tasarımında kullanılması anlatılarak LMS, NLMS ve FG algoritmalarının farklı süzgeç değerlerindeki başarımları karşılaştırılacaktır. Bütün bu karşılaştırmalara ek olarak, SNR değerlerinin ve FG algoritmasının kontrol parametre değişimlerinin AGG tasarımındaki etkiside bu bölümde değerlendirilmiştir.

4.1. Adaptif Süzgeçler

Sayısal haberleşme sistemlerinde, gürültü giderme problemi çok sık olarak karşımıza çıkmakta ve sonuçta elde edilen bilgi işaretinin, gürültüden tamamen arındırılması önemli olmaktadır.

Şekil 4.1'de LMS ve NLMS algoritmaları kullanılarak FIR adaptif gürültü giderici yapısı verilmektedir. Adaptif gürültü gidericide iki giriş vardır: ana giriş, gürültüyle bozulmuş bilgi işaretidir ve referans giriş, ise gürültüyle ilişkili işarettir. Referans girişteki gürültü, giriş işaretinde gürültü elemanlarının bir benzerini elde etmek için adaptif süzgeçle işlenmektedir.



Şekil 4.1. LMS-NLMS algoritması kullanılarak adaptif FIR gürültü giderici.

4.2. FG ALGORİTMASI KULLANILARAK ADAPTİF GÜRÜLTÜ GİDERİCİ TASARIMI

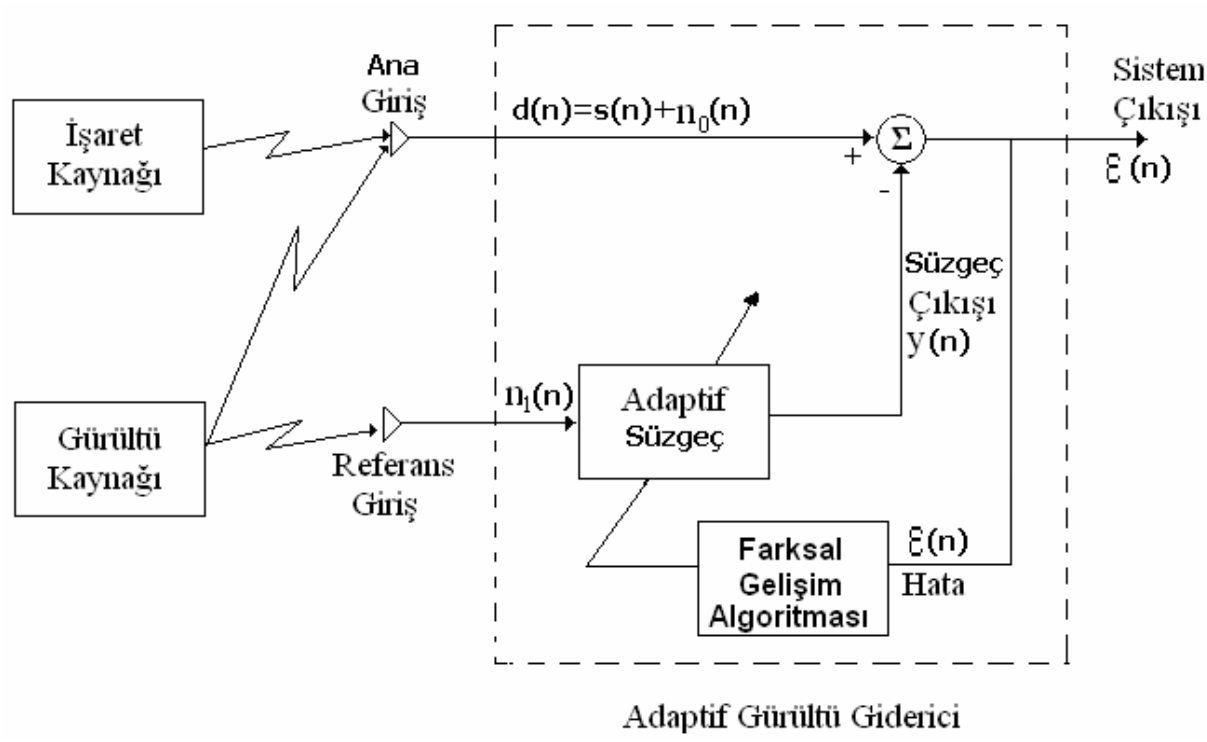
Bu alt bölümde öncelikle, LMS ve NLMS algoritmalarıyla tasarlanan Adaptif gürültü gidericinin başarımları değerlendirilecek ve daha sonra önerilen FG algoritmasının gürültü gidermede nasıl kullanıldığı konusu anlatılarak elde edilen sonuçlar verilecektir.

4.2.1. Tasarım Hakkında Genel Bilgi

Bu çalışmada adaptif gürültü giderici tasarımında literatürde DE/best/2/bin olarak bilinen FG algoritmasına dayalı, bir yapay zeka optimizasyon algoritması önerilmiştir. FG algoritmasının popülasyonu, başlangıç parametre sınırları içerisinde rastgele başlatılmaktadır. Her bir G jenerasyonunda, popülasyon pertürbasyona uğrar. Birbirinden farklı üç birey popülasyondan rastgele seçilir. DE/best/2/bin olarak isimlendirdiğimiz algoritmada popülasyondan rastgele seçilen üç vektörden ikisi arasındaki fark alınıp F ölçekleme faktörüyle adım büyüklüğü ölçeklendirildikten sonra mevcut birey de hesaplamaya katılarak bir deneme vektörü bulunmaktadır. K katsayısı burada rastgele seçilen bir vektörle mevcut birey arasında meydana gelen birleşmenin seviyesini temsil etmektedir. Tipik olarak, eğer yeni birey halen seçili olan bireyden daha iyiyse bu durumda hali hazırdaki birey yeni olanla yer değiştirmektedir. En iyi birey, algoritmada kullanılan hata fonksiyonunun iyilik değeriyle seçilmektedir. Buradaki seçilen en iyi vektör tasarımda kullanılacak süzgeç katsayıları olduğunda yani

rastgele seçilen her bir birey mutasyon ve yeniden birleşme işlemlerine maruz kaldıktan sonra süzgeç katsayılarının oluşumuna yardım etmektedir. Bu çalışmada, adaptif süzgeçlemede yaygın olarak kullanılan ve Eşitlik (2.2)'de verilen MSE hata fonksiyonu kullanılmaktadır. Bu çalışmada, bütün simülasyon uygulamaları için hiçbir bağımlı değişkenle ilişkili olmayan ve beyaz gürültü olarak da adlandırılan ideal gürültü çeşidi olan ve literatürde çokça kullanılan rastgele gürültü kullanılmıştır.

Gelişime dayalı birçok çalışmada yeniden birleşme ve mutasyon değerinin birbirlerinin etkisini arttırdığı gösterilmiştir. FG algoritmasında, yeniden birleşme CR katsayısı ile kontrol edilir. Price 1999'da CR'nin alabileceği değerleri listelemiştir [58]. Buna göre mükemmel fakat yavaş bir optimizasyon için CR'nin değerinin 0.1, hızlı fakat riskli bir yakınsama için 1.0 olarak alınabileceği gösterilmiştir. Burada başlangıç için önerilen CR değeri 0.8'dir. Ayrıca ölçekleme faktörü F katsayısı için Storn ve Price tarafından önerilen değer; $0.4 \leq F \leq 1.0$ arasındadır [18,74]. Cruz ve arkadaşlarının çalışmalarından da görülmektedir ki ölçekleme faktörü $0.4 \leq F \leq 0.6$ arasındaki değerlerde en iyi başarımı sergilemektedir [74]. Bu çalışmada kullanılan F değeri 0.5, birleşme faktörü K 0,4 ve popülasyon büyüklüğü ise 100 olarak alınmıştır [75].



Şekil 4.2. FG algoritmasının AGG'de kullanılması.

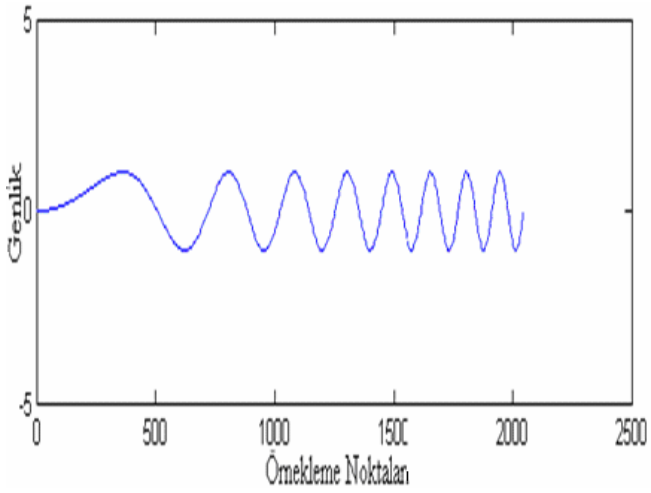
Şekil 4.2'de FG algoritması kullanılarak elde edilen AGG'nin yapısı verilmiştir.

4.2.2. $\sin(2\pi t^2/8N^2)$ Bilgi İşareti Kullanılarak AGG Tasarımı

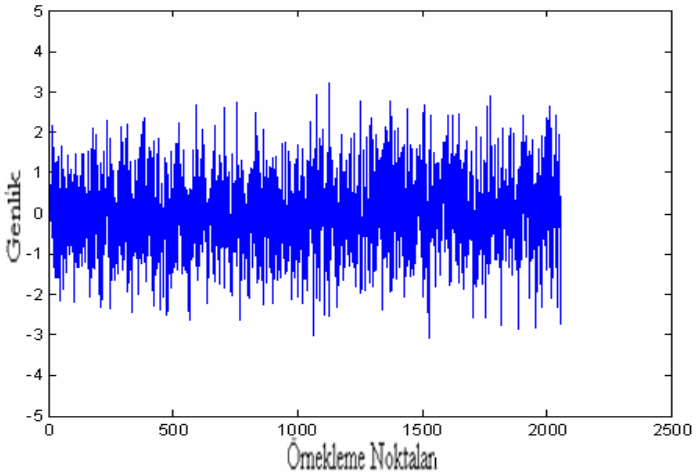
Çalışmanın ilk kısmında, Eşitlik (4.1)'de verilen bilgi işareti kullanılmıştır. Şekil 4.3'de AGG girişi olan tek bir sinüzoidal s(n) bilgi işareti, Şekil 4.4'de $n_0(n)$ gürültü işareti ve Şekil 4.5'de d(n) gürültülü bilgi işareti gösterilmektedir.

$$s(n) = \sin(2 * \pi * t^2 / 8 * N^2) \qquad n = 1,2,.....,1000 \qquad (4.1)$$
$$d(n) = s(n) + n_0(n)$$

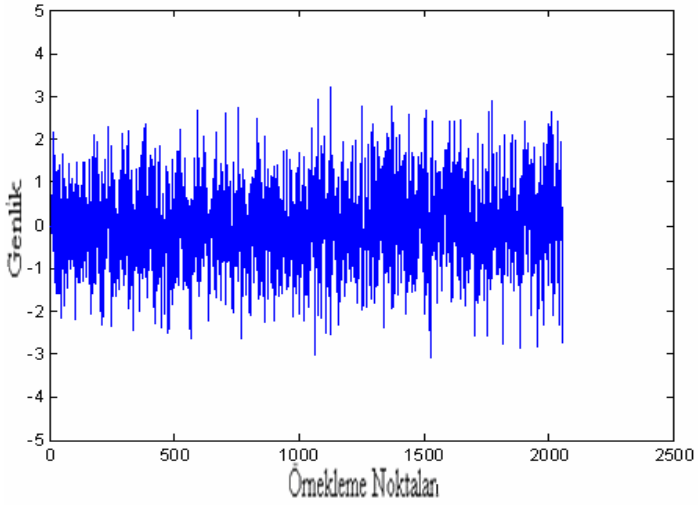
Bu çalışmada kullanılan MSE sıfıra sürülecek hata fonksiyonu, y(n) izlenecek gerçek işaret ve d(n) arzu edilen işarettir.



Şekil 4.3. Bilgi işareti, s(n).



Şekil 4.4. Gürültü işareti, $n_0(n)$.



Şekil 4.5. Gürültülü işaret, d(n).

İlk olarak literatürde çokça denenmiş ve kabul görmüş olan LMS ve NLMS algoritmalarıyla AGG tasarımı gerçekleştirilmiştir. Daha sonra FG algoritmasıyla AGG tasarımı gerçekleştirilerek elde edilen simülasyon sonuçları gösterilmiştir.

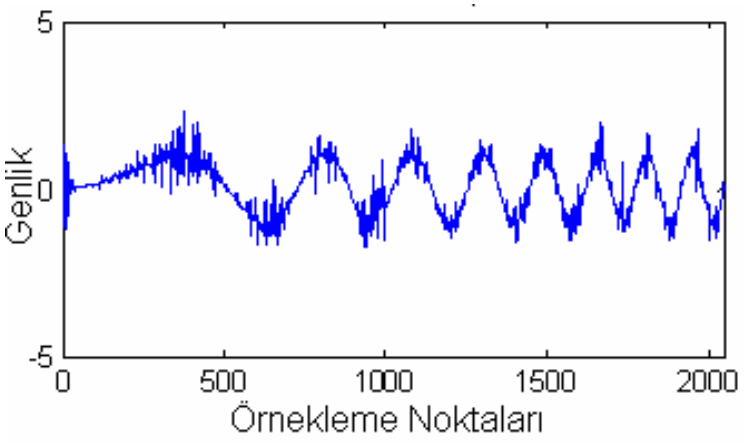
4.2.2.1. LMS Algoritması İle AGG Tasarımı:

Adaptif gürültü giderici tasarımında kullanılan LMS algoritmasının kontrol parametreleri Tablo 4.1'deki gibi alınmıştır. Kullanılan bu kontrol parametreleriyle hata minimize edilmeye çalışılmıştır.

Tablo 4.1. LMS algoritmasının kontrol parametreleri.

Adım büyüklüğü (μ)	*0.1*
İterasyon sayısı	*2048*

Şekil 4.6'da tek bir sinüzoidal bilgi işareti için, LMS algoritması ile elde edilen AGG çıkışı verilmektedir. Şekil 4.6'dan da görüldüğü gibi süzgeç çıkışı arzu edilen işareti yaklaşmayı başarmış olsa da tam olarak yakalayamamaktadır.



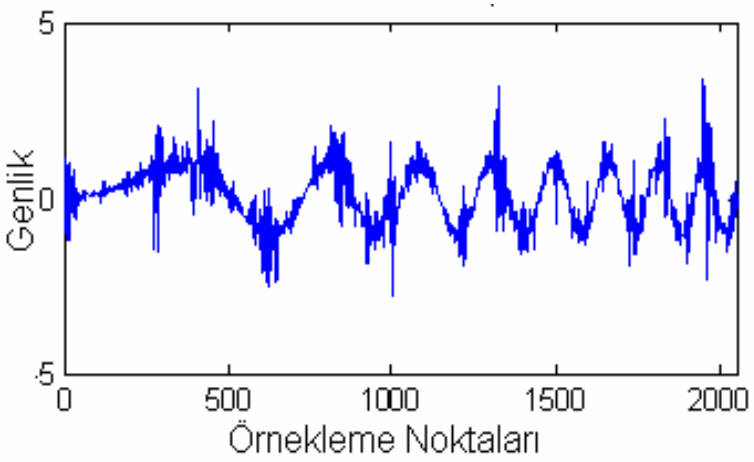
Şekil 4.6. Tek bir sinüzoidal işaret için LMS algoritması kullanıldığında AGG çıkışı.

4.2.2.2. NLMS Algoritması İle AGG Tasarımı:

Tablo 4.2'de bu çalışmadaki NLMS algoritmasının kontrol parametreleri verilmektedir.

Tablo 4.2. NLMS algoritmasının kontrol parametreleri.

Adım büyüklüğü (μ)	*0.1*
İterasyon sayısı	*2048*



Şekil 4.7. Tek bir sinüzoidal işaret için NLMS algoritmasında elde edilen AGG çıkışı.

Şekil 4.7'de tasarlanan AGG çıkışında elde edilen NLMS algoritmasının çıkışı

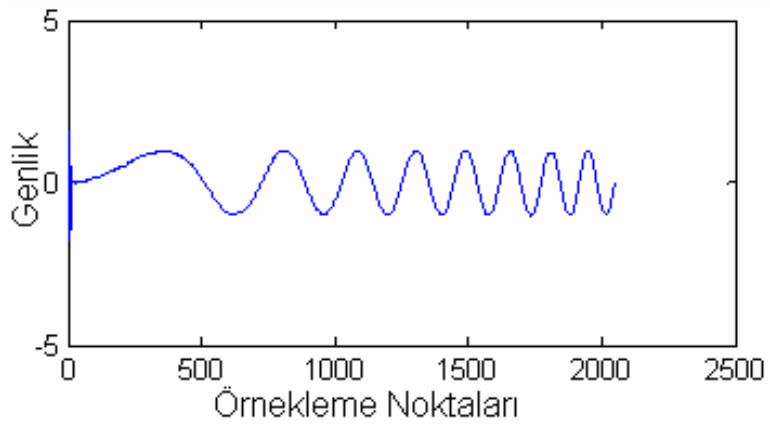
gösterilmektedir. Şekil 4.7 incelenecek olursa NLMS algoritması da süzgeç çıkışı arzu edilen işareti LMS algoritması gibi tam olarak yakalayamamaktadır.

4.2.2.3. FG Algoritması İle AGG Tasarımı:

AGG tasarımda kullanılan FG algoritmasının kontrol parametreleri Tablo 4.3'deki gibi alınmıştır. Şekil 4.8'de önerilen FG algoritmasının, gürültülü işaret olan d(n) işaretinden gürültü bastırıldığında elde edilen, bilgi işareti olan s(n) işaretini yakalamış olduğu görülmektedir.

Tablo 4.3. FG algoritmasının kontrol parametreleri.

Popülasyon büyüklüğü(NP)	100
Çaprazlama oranı (CR)	0,8
Birleşme faktörü (K)	0,4
Ölçekleme faktörü (F)	0,5
İterasyon sayısı (Gmax)	2048



Şekil 4.8. Tek bir sinüzoidal işaret için FG algoritmasında elde edilen AGG çıkışı.

4.2.3. $\sin(2\pi t^2/8N^2)+ \sin(2\pi t^2/4N^2)$ Bilgi İşareti Kullanılarak AGG Tasarımı:

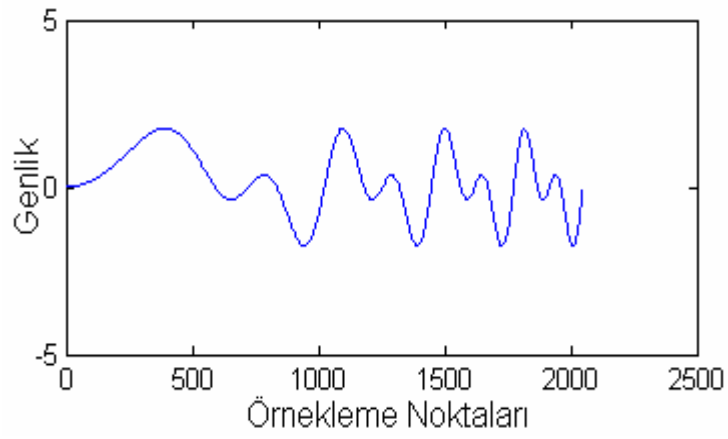
Bu çalışmada kullanılan işaretler, Eşitlik (4.2)'de verilmektedir. Şekil 4.9'da AGG girişi $\sin(2\pi t^2/8N^2)$ ve $\sin(2\pi t^2/4N^2)$ gibi iki sinüzoidal işaret toplamı olan s(n) bilgi işareti, Şekil 4.10'de $n_0(n)$ gürültü işareti ve Şekil 4.11'de ise farklı SNR değerlerinde d(n)

gürültülü bilgi işareti gösterilmektedir.

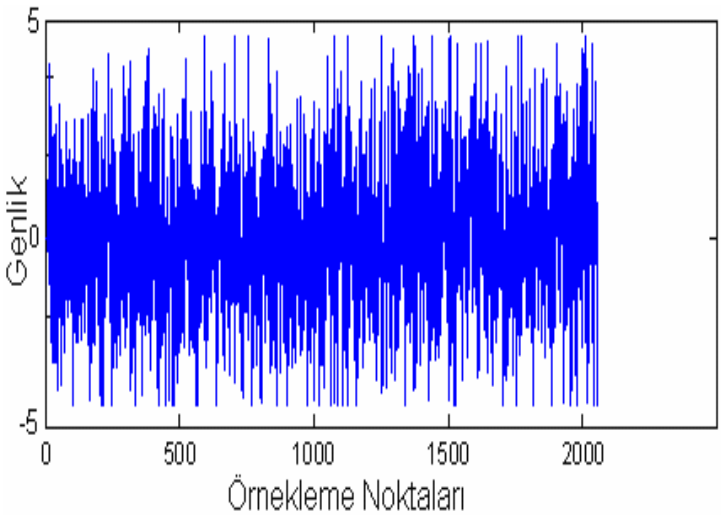
$$s(n) = \sin(2*\pi*t^2/8*N^2) + \sin(2*\pi*t^2/4*N^2) \qquad n = 1,2,.....,1000 \qquad (4.2)$$
$$d(n) = s(n) + n_0(n)$$

Bu çalışmada kullanılan MSE, sıfıra sürülecek hata fonksiyonu, y(n) izlenecek gerçek işaret ve d(n) arzu edilen işarettir.

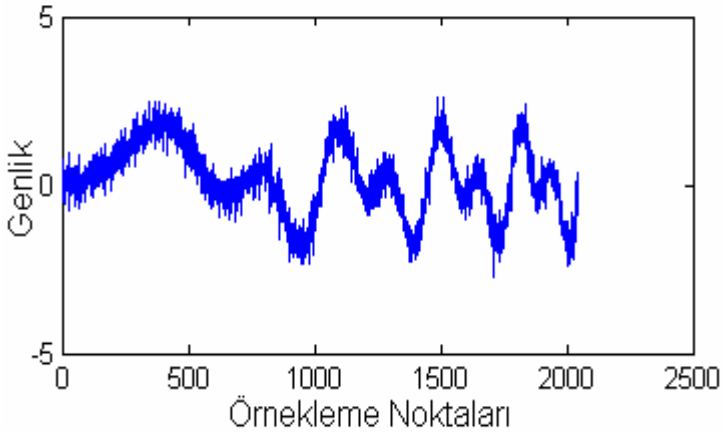
İlk olarak, ilk örneğimizdekine benzer şekilde LMS ve NLMS algoritmalarıyla AGG tasarımı gerçekleştirilmiş, daha sonra FG algoritmasıyla AGG tasarımı gerçekleştirilerek elde edilen simülasyon sonuçları verilmiştir.



Şekil 4.9. Bilgi işareti, s(n).



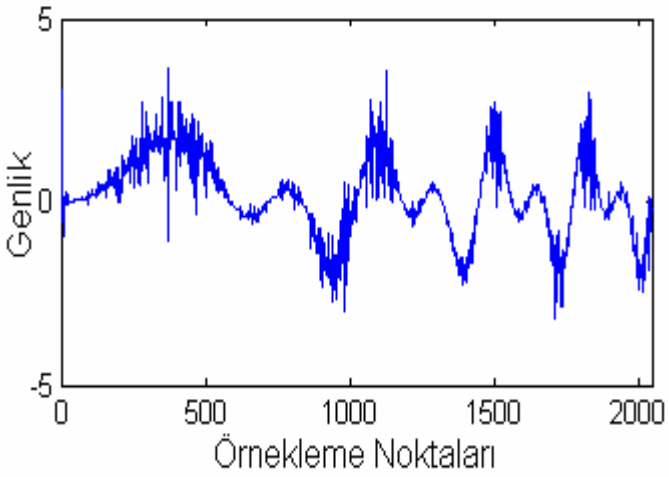
Şekil 4.10. Gürültü işareti, $n_0(n)$.



Şekil 4.11. Gürültülü işaret, d(n).

4.2.3.1. LMS Algoritması İle AGG Tasarımı:

AGG tasarımında kullanılan LMS algoritmasının kontrol parametreleri Tablo 4.1'deki gibi alınmıştır.

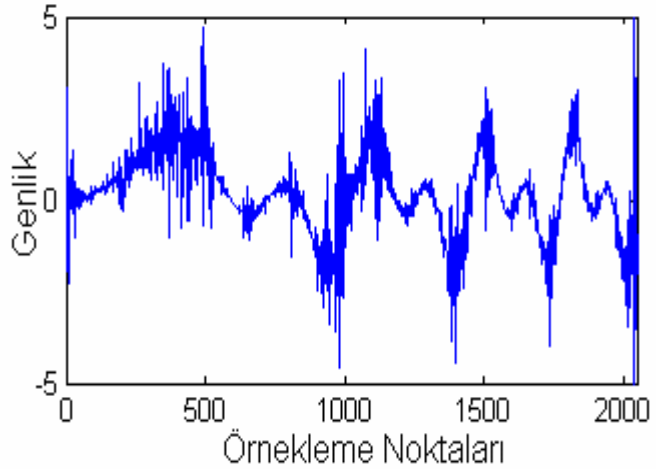


Şekil 4.12. İki sinüzoidal işaret toplamında LMS algoritmasında elde edilen AGG çıkışı.

Kullanılan bu kontrol parametreleriyle hata, minimize edilmeye çalışılmıştır. Şekil 4.12'de iki sinüzoidal işaretin toplamı olan bilgi işareti için, LMS algoritması ile elde edilen AGG çıkışı verilmektedir. Şekil 4.12 incelenecek olursa süzgeç çıkışı, arzu edilen işareti yaklaşmayı başarmış olsa da tam olarak yakalayamamaktadır.

4.2.3.2. NLMS Algoritması İle AGG Tasarımı:

Tablo 4.2'de alınan NLMS algoritmasının kontrol parametreleri kullanılmıştır.

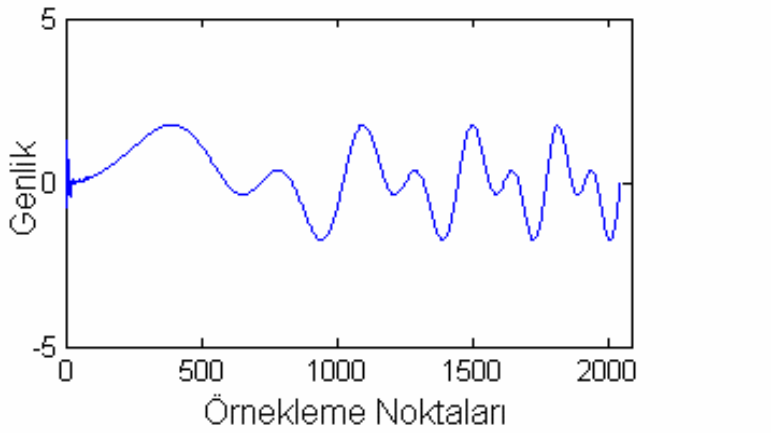


Şekil 4.13. İki sinüzoidal işaret toplamında NLMS algoritmasında elde edilen AGG çıkışı.

Şekil 4.13'te NLMS algoritması ile tasarlanan AGG'nin çıkışı gösterilmektedir. Şekil 4.13'ten de görüldüğü gibi NLMS algoritması kullanıldığında da süzgeç çıkışı arzu edilen işareti LMS algoritması gibi tam olarak yakalayamadığı görülmektedir.

4.2.3.3. FG Algoritması İle AGG Tasarımı:

Çalışmada AGG tasarımda kullanılan FG algoritmasının kontrol parametreleri, Tablo 4.3'deki gibi alınmıştır.



Şekil 4.14. İki sinüzoidal işaret toplamında FG algoritmasında elde edilen AGG çıkışı.

Şekil 4.14'de önerilen FG algoritmasının, gürültülü işaret olan d(n) işaretinden gürültü bastırıldığında elde edilen iki sinüzoidal işaretinin toplamı olan bilgi işareti olan s(n) işaretini yakalamış olduğu görülmektedir.

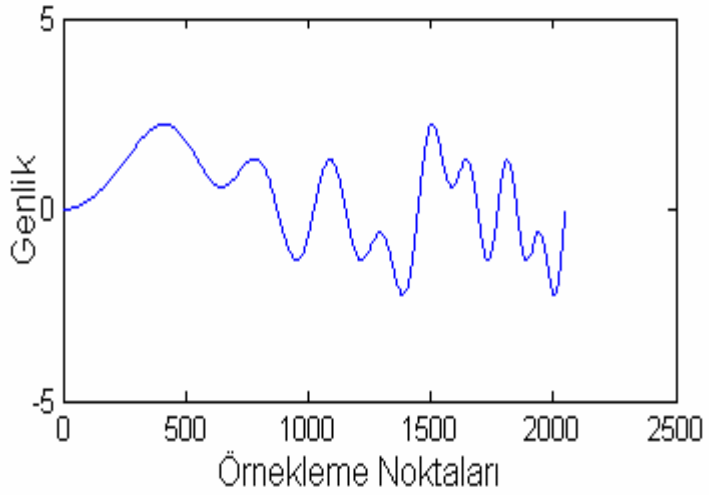
4.2.4. $\sin(2\pi t^2/8N^2)+\sin(2\pi t^2/4N^2)+\sin(2\pi t^2/2N^2)$ Bilgi İşareti Kullanılarak AGG Tasarımı:

Bu çalışmada kullanılan işaretler, Eşitlik (4.3)'te verilmektedir. Şekil 4.15'de AGG, girişi $\sin(2\pi t^2/8N^2)$, $\sin(2\pi t^2/4N^2)$ ve $\sin(2\pi t^2/2N^2)$ olan üç sinüzoidal işaret toplamı olan s(n) bilgi işareti, Şekil 4.16'da $n_0(n)$ gürültü işareti ve Şekil 4.17'de ise farklı SNR değerlerinde d(n) gürültülü bilgi işareti gösterilmektedir.

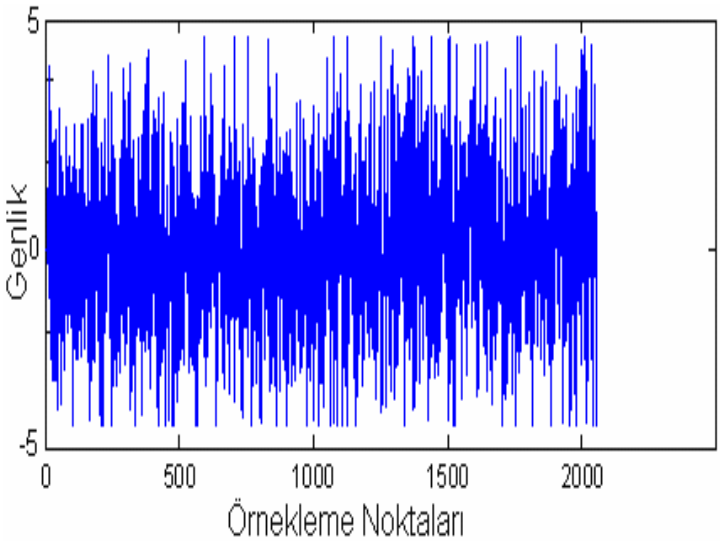
$$s(n) = \sin(2*\pi*t^2/8*N^2) + \sin(2*\pi*t^2/4*N^2) + \sin(2*\pi*t^2/2*N^2) \quad (4.3)$$
$$d(n) = s(n) + n_0(n)$$
$$n = 1,2,.....,1000$$

Bu çalışmada kullanılan MSE sıfıra sürülecek hata fonksiyonu, y(n) izlenecek gerçek işaret ve d(n) arzu edilen işarettir.

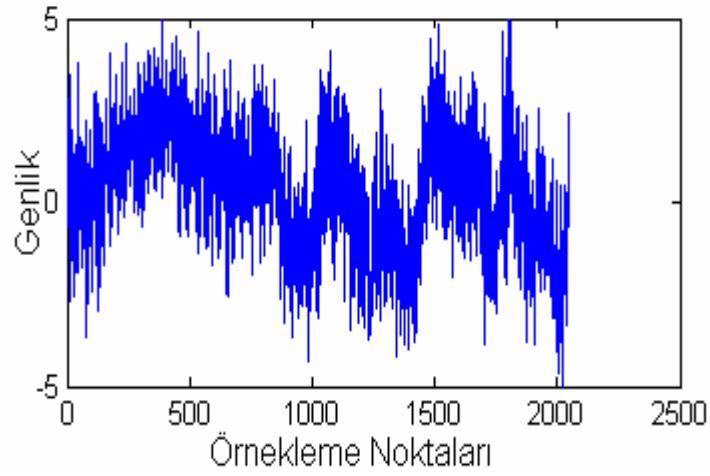
İlk olarak yine LMS ve NLMS algoritmasıyla AGG tasarımı gerçekleştirilmiştir. Daha sonra FG algoritmasıyla AGG tasarımı gerçekleştirilerek elde edilen simülasyon sonuçları verilmiştir.



Şekil 4.15. Bilgi işareti, s(n).



Şekil 4.16. Gürültü işareti, $n_0(n)$.

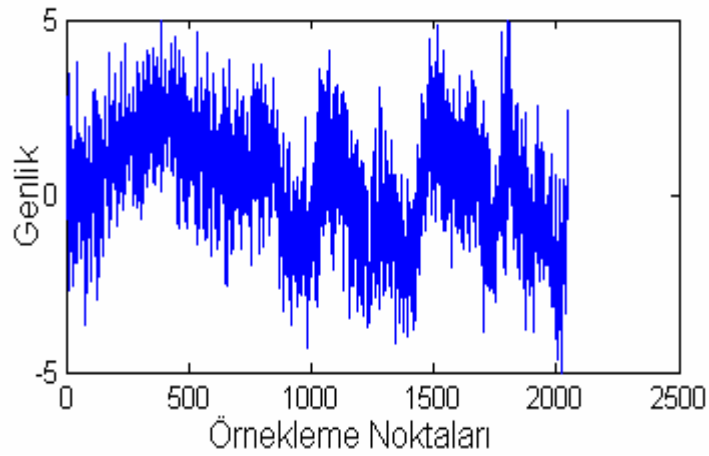


Şekil 4.17. Gürültülü işaret, d(n).

4.2.4.1. LMS Algoritması İle AGG Tasarımı:

AGG tasarımında kullanılan LMS algoritmasının kontrol parametreleri Tablo 4.1'deki gibi alınmıştır.

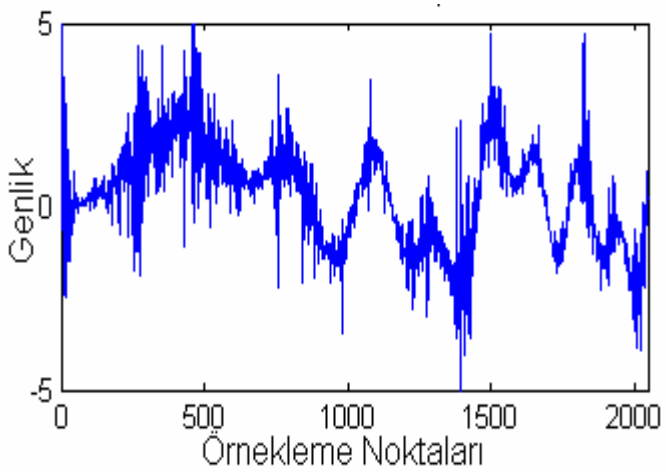
Kullanılan bu kontrol parametreleriyle hata, minimize edilmeye çalışılmıştır. LMS algoritması kullanıldığında elde edilen AGG'nin çıkışı, Şekil 4.18 ile verilmektedir. Şekil 4.18'den de görüldüğü gibi süzgeç çıkışı arzu edilen işareti yaklaşmayı başarmış olsa da tam olarak yakalayamamaktadır.



Şekil 4.18. Üç sinüzoidal işaret toplamında LMS algoritmasında elde edilen AGG çıkışı.

4.2.4.2. NLMS Algoritması İle AGG Tasarımı:

AGG tasarımında kullanılan NLMS algoritmasının kontrol parametreleri Tablo 4.2'deki gibi alınmıştır.



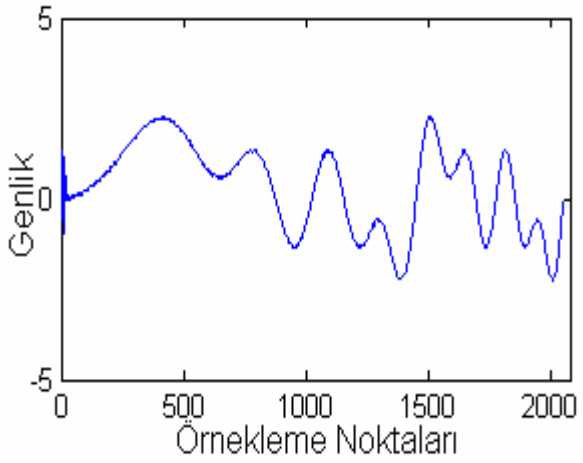
Şekil 4.19. Üç sinüzoidal işaret toplamında NLMS algoritmasında elde edilen AGG çıkışı.

Şekil 4.19'da tasarlanan AGG çıkışında elde edilen NLMS algoritmasının çıkışı gösterilmektedir. Şekil 4.19'dan da görüldüğü gibi üç sinüzoidal işaretin toplamı olan bilgi işareti için, NLMS algoritması da süzgeç çıkışı arzu edilen işareti LMS algoritmasında olduğu gibi tam olarak yakalayamamıştır.

4.2.4.3. FG Algoritması İle AGG Tasarımı:

AGG tasarımında kullanılan FG algoritmasının kontrol parametreleri, daha önce kullanılan Tablo 4.3'deki gibi alınmıştır.

Şekil 4.20'de önerilen FG algoritmasının gürültülü işaret olan d(n) işaretinden gürültü bastırıldığında elde edilen iki sinüzoidal işaretin toplamı olan bir bilgi işareti olan s(n) işaretini yakalamış olduğu görülmektedir.



Şekil 4.20. Üç sinüzoidal işaret toplamında FG algoritmasında elde edilen AGG çıkışı.

4.3. FARKLI SÜZGEÇ DERECELERİ İÇİN AGG TASARIMI

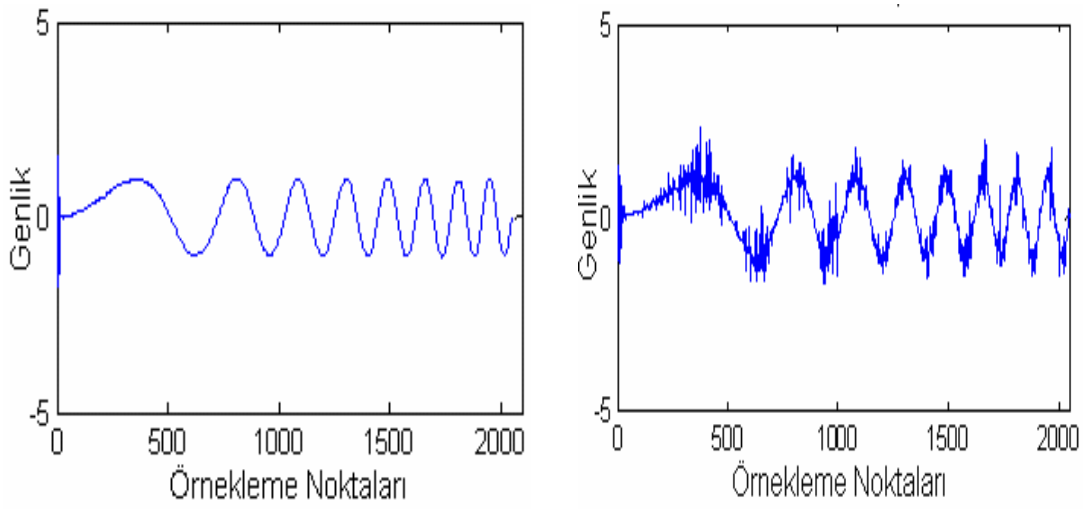
Çalışmanın bu bölümünde yine üç farklı sinüzoidal işaretle, farklı süzgeç dereceleri için AGG tasarımında LMS, NLMS ve FG algoritmalarının başarımları incelenecektir.

4.3.1. $\sin(2\pi t^2/8N^2)$ Bilgi İşareti Kullanılarak Farklı Süzgeç Dereceleri ile AGG Tasarımı:

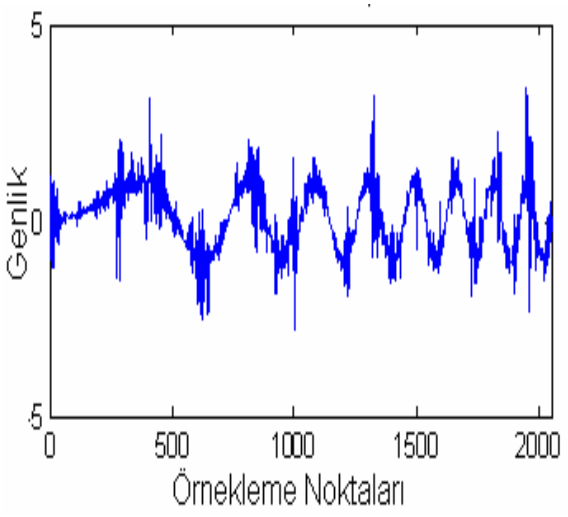
Bu bölümde farklı bilgi işaretleri için farklı süzgeç dereceleri kullanıldığında AGG'nin başarımının ölçülmesi hedeflenmiştir. İlk olarak $\sin(2\pi t^2/8N^2)$ bilgi işareti için iki, dört, altı, sekiz ve onuncu dereceden süzgeç kullanılarak AGG'nin çıkışları karşılaştırılacaktır.

4.3.1.1. İkinci Dereceden Süzgeç Kullanılarak AGG:

Bu bölümde Tablo 4.1'de verilen LMS ve NLMS algoritması için kullanılan kontrol parametreleri ve Tablo 4.3'de verilen FG algoritmasının kontrol parametreleri incelediğimiz ilk işaret için kullanılmaktadır.



(a) (b)



(c)

Şekil 4.21. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi iki olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

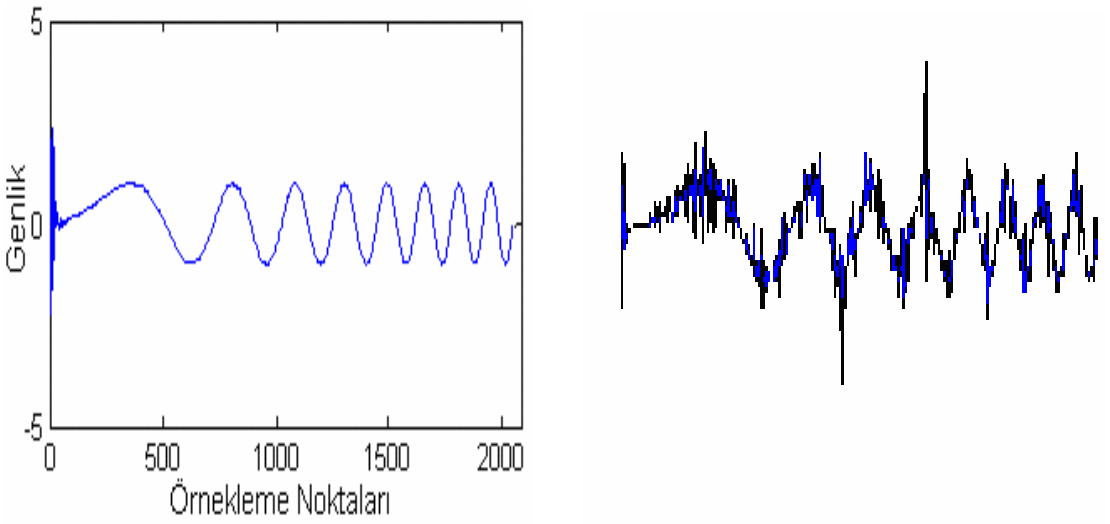
Şekil 4.21'de süzgeç derecesi iki olduğu zaman FG, LMS ve NLMS algoritmasının AGG çıkışı ve Tablo 4.4'de ise FG, LMS ve NLMS algoritmaları kullanıldığında elde edilen katsayılar verilmektedir.

Tablo 4.4. FG, LMS ve NLMS algoritmaları ile ikinci dereceden süzgeç için elde edilen katsayılar.

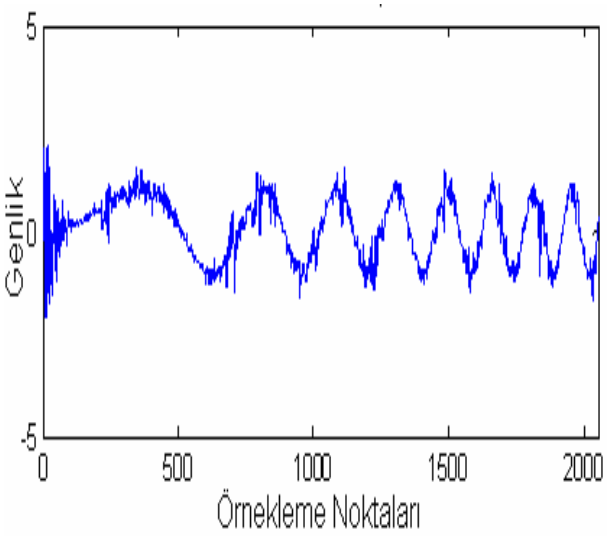
Katsayılar	FG	LMS	NLMS
W1	-0.4296	-0.3901	-0.3759
W2	1.0566	1.0860	1.0705

Şekil 4.21'de süzgeç derecesi iki olduğu zaman elde edilen FG algoritmasının AGG çıkışını, diğer iki klasik algoritmadan elde edilen AGG'nin çıkışı ile karşılaştırdığımızda FG algoritmasının başarılı bir şekilde gürültüyü bastırdığı ve bilgi işaretini yeniden elde edebildiği görülmektedir.

4.3.1.2. Dördüncü Dereceden Süzgeç Kullanılarak AGG:



(a) (b)



(c)

Şekil 4.22. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi dört olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Çalışmanın bu kısmında, süzgeç derecesi dört olduğu zaman AGG'nin FG, LMS ve NLMS algoritmaları için gösterdikleri başarımlar karşılaştırılmıştır. Şekil 4.22'de süzgeç derecesi dört olduğu zaman FG, LMS ve NLMS algoritmasının AGG çıkışı ve Tablo 4.5'de FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayı değerleri verilmektedir.

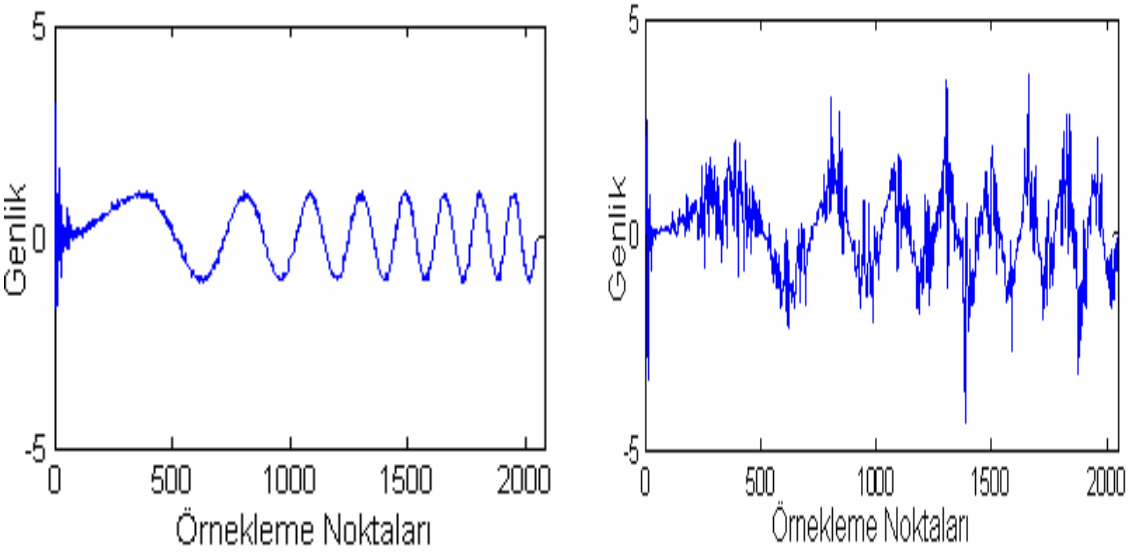
Tablo 4.5. FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayılar.

Katsayılar	**FG**	**LMS**	**NLMS**
W1	0.1483	0.1311	0.2362
W2	-0.5389	-0.5532	-0.4063
W3	0.7321	0.7155	0.7743
W4	0.8808	0.8438	0.8862

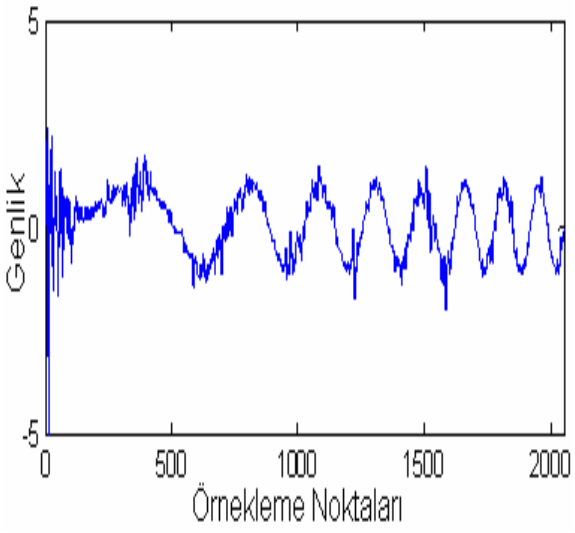
Şekil 4.22'de süzgeç derecesi dört olduğu zaman elde edilen FG algoritmasının AGG çıkışı, LMS ve NLMS algoritmalarının AGG çıkışı ile karşılaştırdığımızda, FG algoritmasının başarılı bir şekilde gürültüyü bastırabilmiştir.

4.3.1.3. Altıncı Dereceden Süzgeç Kullanılarak AGG:

Şekil 4.23'de süzgeç derecesi altı olduğu zaman FG, LMS ve NLMS algoritmasının AGG çıkışı ve Tablo 4.6'da FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar gösterilmektedir.



(a) (b)



(c)

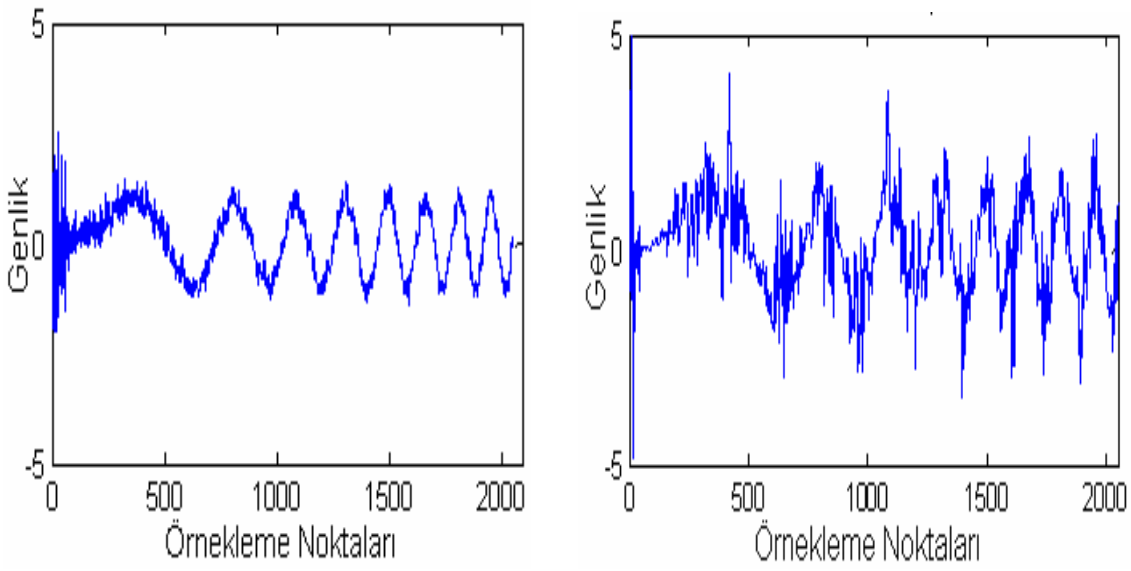
Şekil 4.23. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi altı olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Şekil 4.23'de süzgeç derecesi altı olduğu zaman elde edilen FG algoritmasının AGG çıkışını, LMS ve NLMS algoritmalarının AGG çıkışı ile karşılaştırdığımızda, FG algoritmasının gürültüyü gidermede iyi bir başarıma sahip olduğu söylenebilmektedir.

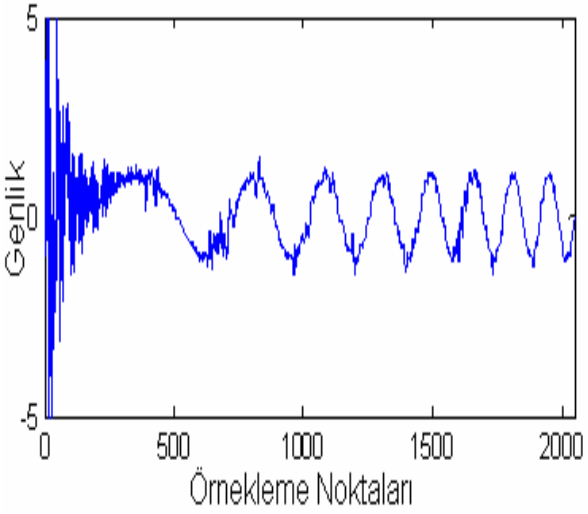
Tablo 4.6. FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar.

Katsayılar	FG	LMS	NLMS
W1	-0.4288	-0.4461	-0.1557
W2	-0.0083	-0.0172	0.1802
W3	0.1236	0.1286	0.2749
W4	-0.0010	0.0352	0.1975
W5	0.6428	0.6576	0.7193
W6	0.9794	1.0054	1.0346

4.3.1.4. Sekizinci Dereceden Süzgeç Kullanılarak AGG:



(a) (b)



(c)

Şekil 4.24. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi sekiz olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Süzgeç derecesi sekiz olduğu zaman FG, LMS ve NLMS algoritmasının AGG çıkışı karşılaştırılmış ve Şekil 4.24'de bu algoritmaların adaptif gürültü giderici çıkışları gösterilmiştir.

Tablo 4.7. FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç için elde edilen katsayılar.

Katsayılar	**FG**	**LMS**	**NLMS**
W1	-0.3080	-0.3123	0.2129
W2	0.8863	0.7836	0.5746
W3	-1.1582	-1.1924	-0.5534
W4	-0.4338	-0.5396	-0.1702
W5	1.9451	2.1622	1.4534
W6	-1.1271	-1.1953	-0.5791
W7	-0.8069	-0.7312	-0.5638
W8	-0.7851	-0.7675	-0.5336

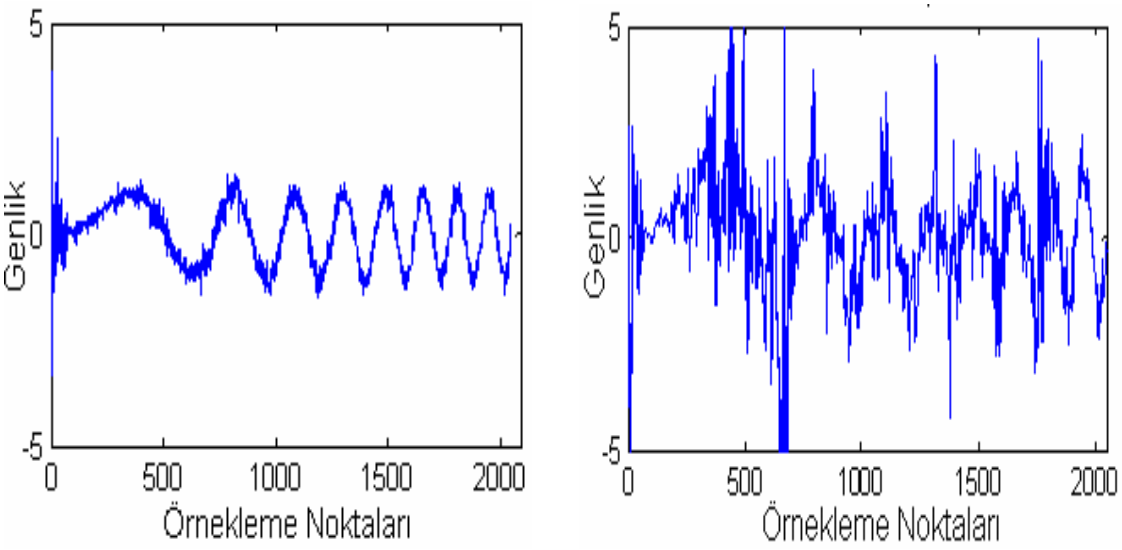
Tablo 4.7'de ise FG, LMS ve NLMS algoritmaları kullanıldığında sekizinci dereceden süzgeç için elde edilen katsayılar verilmektedir. Şekil 4.24'de süzgeç derecesi sekiz

olduğu zaman elde edilen LMS ve NLMS algoritmalarının AGG çıkışı FG algoritmasının AGG çıkışı ile karşılaştırdığımızda FG algoritmasının gürültüyü gidermede diğer algoritmalara göre daha başarılı olduğu görülmektedir.

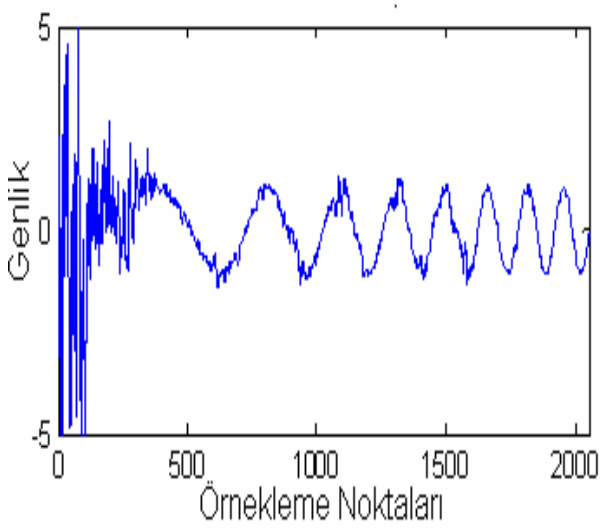
4.3.1.5. Onuncu Dereceden Süzgeç Kullanılarak AGG:

Süzgeç derecesi on olduğu zaman FG, LMS ve NLMS algoritmasının Adaptif gürültü giderici çıkışı karşılaştırılmış ve Şekil 4.25'de bu algoritmaların adaptif gürültü giderici çıkışları gösterilmiştir. Tablo 4.8'de ise FG, LMS ve NLMS algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar verilmektedir.

Şekil 4.25'de süzgecin derecesi on olduğu zaman elde edilen LMS ve NLMS algoritmalarının AGG çıkışını, FG algoritmasının çıkışı ile karşılaştırdığımızda FG algoritmasının gürültüyü, önceki incelediğimiz farklı dereceden süzgeçlerde olduğu gibi, daha iyi bir başarım sergileyerek giderdiği görülmüştür.



(a) (b)



(c)

Şekil 4.25. Tek sinüzoidal bilgi işaretinde ve süzgeç derecesi on olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Tablo 4.8. FG, LMS ve NLMS algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar.

Katsayılar	**FG**	**LMS**	**NLMS**
W1	-0.7098	-1.1269	-0.1305
W2	-0.5119	-0.9635	-0.0686
W3	-0.1403	-0.1908	0.3597
W4	-0.2618	-0.2650	0.2155
W5	-0.4218	-0.2711	0.3392
W6	0.5493	0.0607	0.6221
W7	-0.4963	-0.2944	0.3144
W8	-0.4175	-0.2973	0.4652
W9	0.8118	0.7801	1.1196
W10	-0.3453	-0.5891	-0.1310

Şekil 4.21, Şekil 4.22, Şekil 4.23, Şekil 4.24 ve Şekil 4.25 incelenecek olursa süzgeç derecesi arttırıldığında da önerdiğimiz algoritma, gürültüyü gidermede başarılı

olmaktadır. İncelediğimiz iki klasik algoritma göz önüne alındığında, gürültüyü bastırma işleminde en başarılı olan algoritma; önerdiğimiz, bir yapay zeka optimizasyon algoritması olan FG algoritması olmuştur.

Çalışmanın başlangıç bölümünde, tek bir bilgi işareti kullanıldığı durumda farklı süzgeç dereceleri için LMS, NLMS algoritmaları ve önerilen FG algoritmasının gürültüyü etkin bir şekilde bastırdığı görülmüştür.

Tablo 4.9. $\sin(2\pi t^2/8N^2)$ bilgi işareti için farklı süzgeç derecelerinde SNR değerleri.

Süzgeç derecesi	**SNR(dB)**
2	-2.2666
4	-2.8181
6	-2.7553
8	-6.4378
10	-4.5575

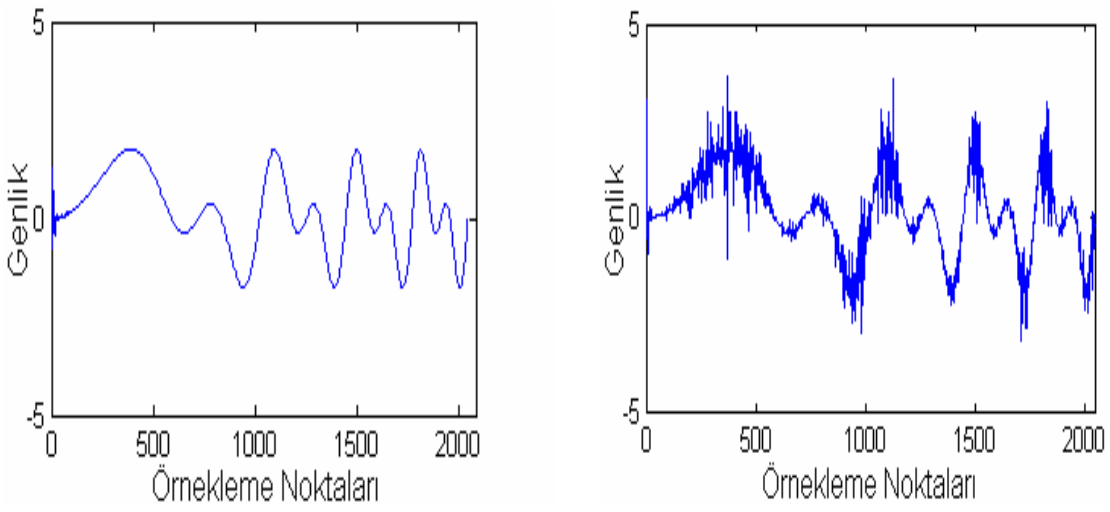
Tablo 4.9'da farklı süzgeç dereceleri için FG, LMS ve NLMS algoritmalarının SNR değerleri verilmektedir. Tablo 4.9'da görüldüğü gibi süzgeç derecesi iki, dört ve altı olduğunda sırasıyla -2.2666 dB, -2.8181 dB ve -2.7553 dB gibi birbirine yakın SNR değerlerinde ve süzgeç derecesinin sekiz ve on olduğu zaman ise SNR değerinin sırasıyla, 6.4378 dB ve -4.5575 dB gibi farklı artışlar şeklinde verildiği durumlarda bile gürültü gidericini başarımının yüksek olduğu görülmüştür.

4.3.2. $\sin(2\pi t^2/8N^2)+\sin(2\pi t^2/4N^2)$ Bilgi İşareti Kullanılarak Farklı Süzgeç Dereceleri ile AGG Tasarımı:

Bu bölümde farklı bir bilgi işareti ve farklı süzgeç dereceleri kullanılarak AGG'nin başarımını ölçmek için ikinci bir bilgi işareti kullanılmıştır. Bunun için; $\sin(2\pi t^2/8N^2)$ ve $\sin(2\pi t^2/4N^2)$ gibi iki işaretin toplamı olan bir bilgi işareti alınmış ve yine önceki işarette olduğu gibi, farklı süzgeç dereceleri için AGG'nin çıkışları karşılaştırılmıştır.

4.3.2.1. İkinci Dereceden Süzgeç Kullanılarak AGG:

Bu bölümde Tablo 4.1'de verilen LMS ve NLMS algoritması için kullanılan kontrol parametreleri ve Tablo 4.3'de verilen FG algoritmasının kontrol parametreleri incelediğimiz ikinci işaret için kullanılmaktadır.



Şekil 4.26. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi 2 olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Şekil 4.26'da süzgeç derecesinin iki olduğu durumda, FG, LMS ve NLMS algoritmaları kullanıldığındaki AGG çıkışı ve Tablo 4.10'da ise yine bu algoritmalar kullanıldığı

durumda elde edilen süzgeç katsayıları verilmektedir.

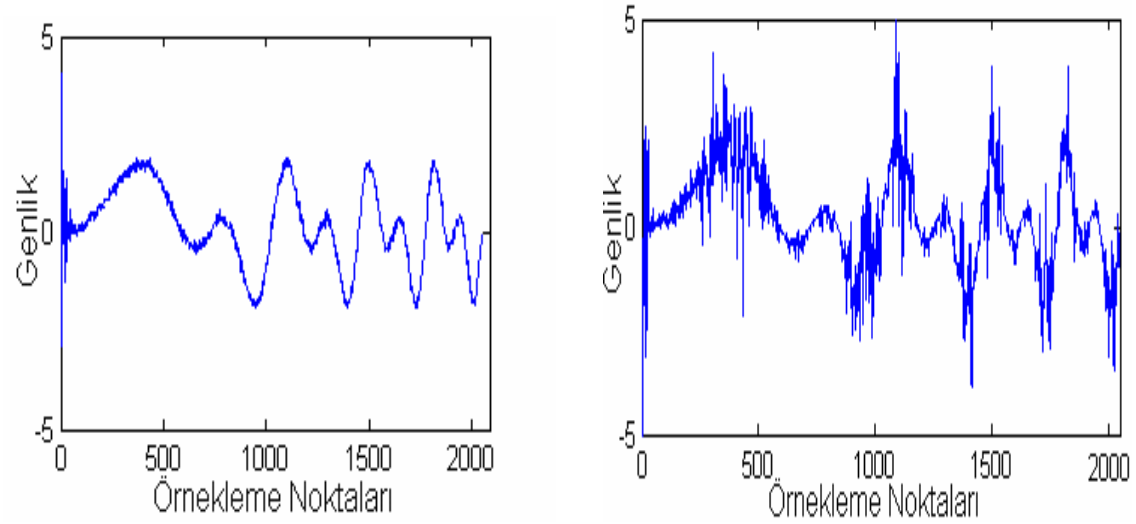
Tablo 4.10. FG, LMS ve NLMS algoritmaları ile ikinci dereceden süzgeç için elde edilen katsayılar.

Katsayılar	FG	LMS	NLMS
W1	-0.3709	-0.3550	-0.3261
W2	1.5773	1.6048	1.6148

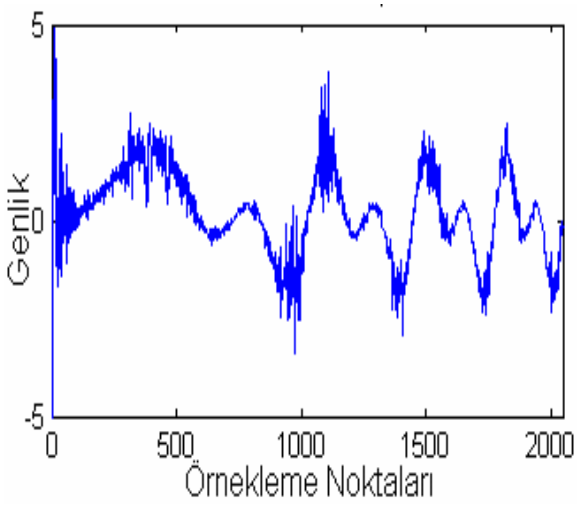
Şekil 4.26'da ikinci dereceden süzgeç kullanıldığında, FG algoritmasının AGG çıkışını diğer iki klasik algoritmanın AGG çıkışı ile karşılaştırdığımızda, FG algoritmasının, başarılı bir şekilde gürültüyü bastırdığı ve bilgi işaretini yeniden elde edebildiği görülmektedir.

4.3.2.2. Dördüncü Dereceden Süzgeç Kullanılarak AGG:

Dördüncü dereceden süzgeç kullanıldığı zaman AGG'nin FG, LMS ve NLMS algoritmaları için gösterdikleri başarımlar, bu kısımda karşılaştırılmıştır.



(a) (b)



(c)

Şekil 4.27. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi dört olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Şekil 4.27'de dördüncü dereceden süzgeç için FG, LMS ve NLMS algoritmasının AGG çıkışı ve Tablo 4.11'de ise bu çalışmada elde edilen süzgeç katsayıları verilmektedir. Yine dördüncü dereceden süzgeç kullanıldığı durumda da, FG algoritmasının başarılı bir şekilde gürültüyü bastırdığı görülmüştür.

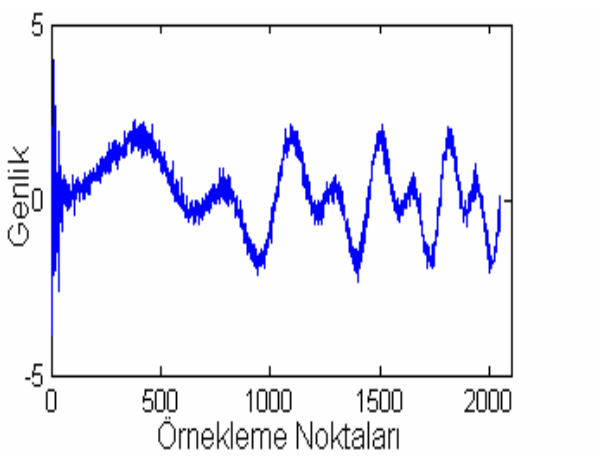
Tablo 4.11. FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayılar.

Katsayılar	**FG**	**LMS**	**NLMS**
W1	-0.6610	-0.7044	-0.4884
W2	0.0038	-0.0159	0.0404
W3	-1.9655	-1.9689	-1.7097
W4	-0.8432	-0.8744	-0.7938

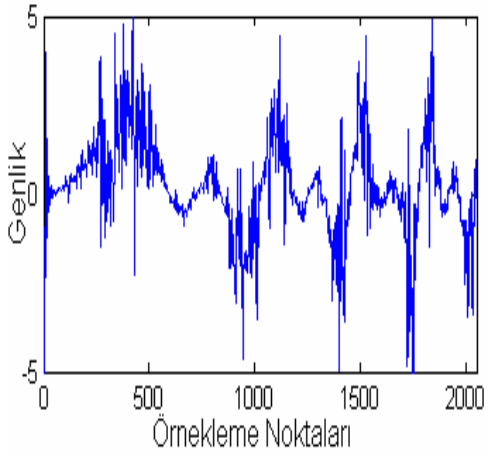
4.3.2.3. Altıncı Dereceden Süzgeç Kullanılarak AGG:

Şekil 4.28'de süzgecin derecesinin altı olduğu durumda, FG, LMS ve NLMS algoritmalarının AGG çıkışı ve Tablo 4.12'de ise bu algoritmalar kullanıldığında elde

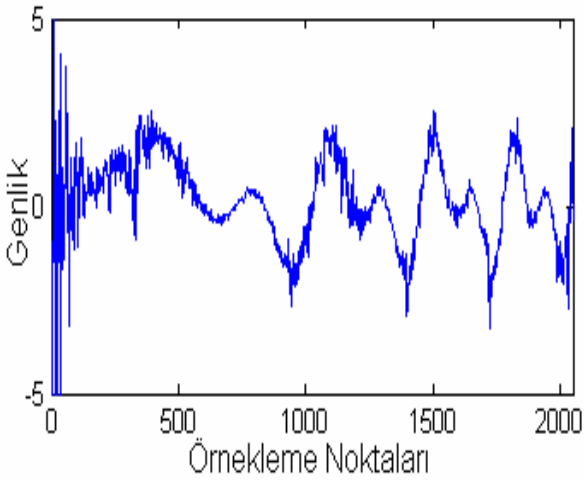
edilen süzgeç katsayıları gösterilmektedir.



(a)



(b)



(c)

Şekil 4.28. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi altı olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

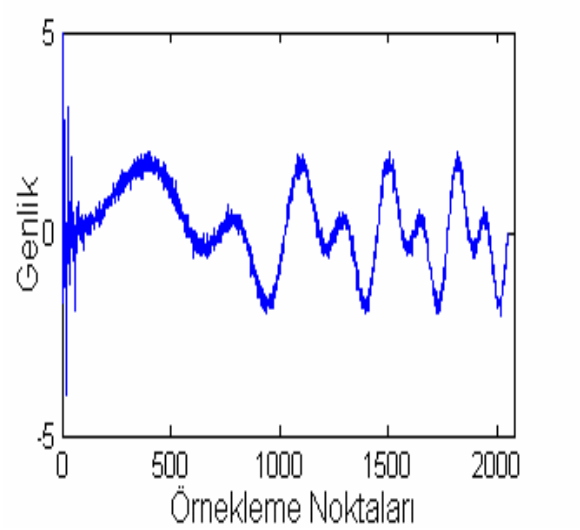
Yine bu kısımda yaptığımız çalışmada da yani, altıncı dereceden süzgeç kullanıldığı durumda da, FG algoritmasının üç algoritma içerisinden daha iyi bir başarım sergileyerek gürültüyü giderdiği görülmüştür.

Tablo 4.12. FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar.

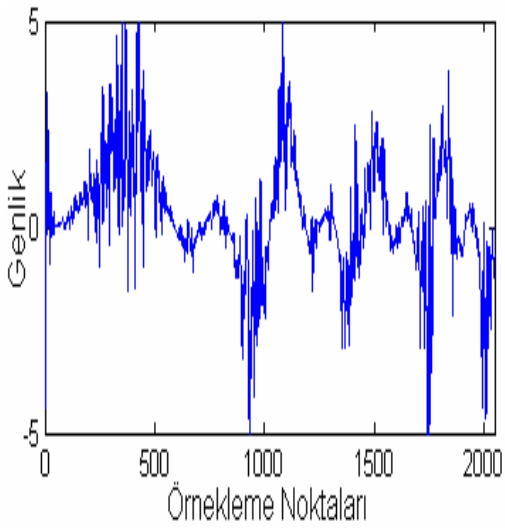
Katsayılar	FG	LMS	NLMS
W1	-0.1831	-0.1807	-0.0368
W2	-1.4497	-1.5390	-1.0905
W3	-2.1004	-2.2310	-1.7020
W4	-0.5105	-0.6593	-0.3960
W5	0.4569	0.3839	0.5019
W6	-0.2521	-0.2117	0.0362

4.3.2.4. Sekizinci Dereceden Süzgeç Kullanılarak AGG:

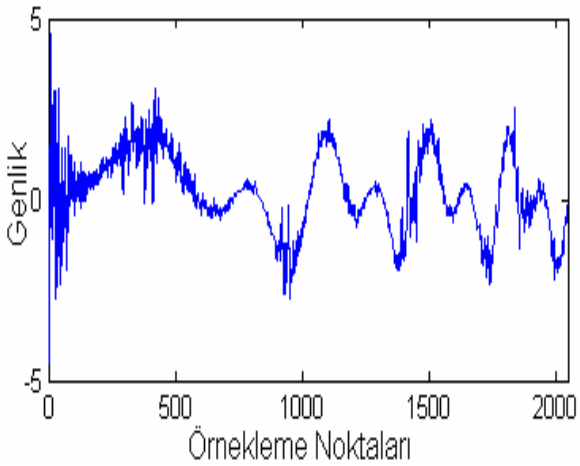
Süzgecin derecesi sekiz olduğu zaman FG, LMS ve NLMS algoritmasının AGG çıkışı karşılaştırılmış ve Şekil 4.29'da bu algoritmaların AGG çıkışları gösterilmiştir.



(a)



(b)



(c)

Şekil 4.29. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi sekiz olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

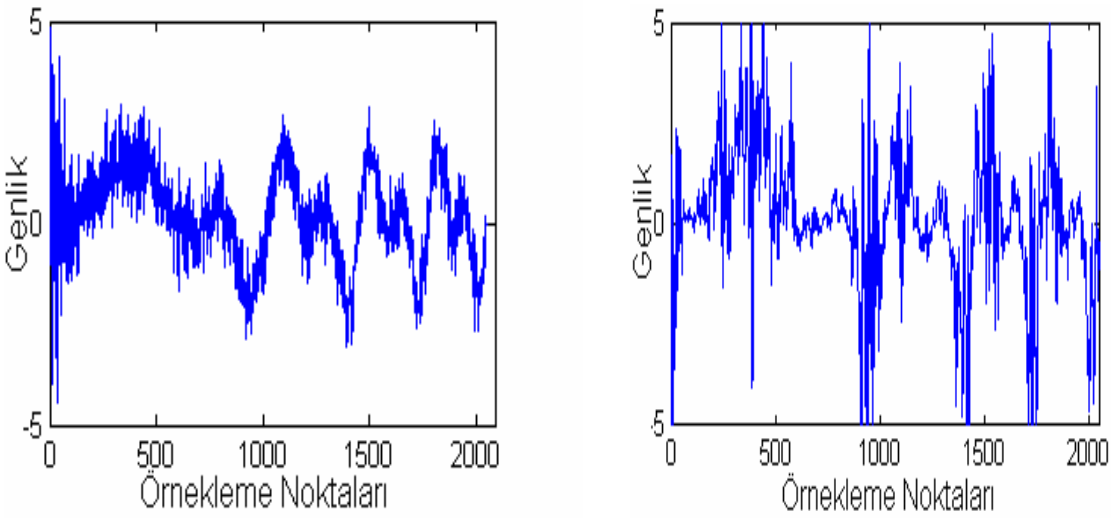
Süzgeç derecesi sekiz olduğu zamanda, FG algoritmasının gürültüyü gidermede LMS ve NLMS'ye oranla daha hızlı ve başarılı olduğu görülmüştür. Tablo 4.13'de FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç kullanıldığında elde edilen katsayılar verilmektedir.

Tablo 4.13. FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç için elde edilen katsayılar.

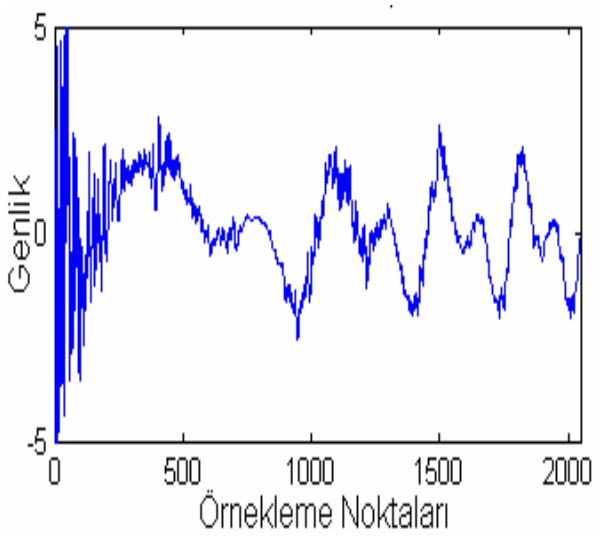
Katsayılar	**FG**	**LMS**	**NLMS**
W1	-0.5136	-0.7843	-0.1138
W2	0.7017	0.5912	0.7195
W3	0.3618	0.3623	0.3992
W4	0.2866	0.3212	0.4498
W5	0.1172	0.1087	0.2986
W6	0.5995	0.5082	0.4700
W7	1.3786	1.6629	1.2688
W8	-1.3190	-1.3976	-0.8543

4.3.2.5. Onuncu Dereceden Süzgeç Kullanılarak AGG:

Çalışmanın bu kısmında da, süzgeç derecesi on olduğunda FG, LMS ve NLMS algoritmalarının AGG çıkışları karşılaştırılmıştır. Şekil 4.30'da bu algoritmaların AGG çıkışları gösterilmiştir.



(a) (b)



(c)

Şekil 4.30. İki sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi on olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Tablo 4.14. FG, LMS ve NLMS algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar.

Katsayılar	FG	LMS	NLMS
W1	-0.0780	-0.0699	0.3602
W2	-0.1844	-0.1242	0.3362
W3	0.1803	-0.3903	0.3309
W4	-1.5543	-1.6218	-0.7384
W5	-0.5151	-0.5245	-0.0223
W6	0.8546	1.2433	1.3547
W7	0.4502	0.4704	0.6767
W8	-0.7949	-0.7111	-0.1659
W9	1.0564	1.1950	1.1881
W10	1.0061	1.4504	1.4235

Onuncu dereceden süzgeç kullanıldığında yine üç farklı algoritma ile elde edilen süzgeç katsayıları, Tablo 4.14'te verilmektedir. Süzgeç derecesi on olduğu durumda da, FG algoritmasının AGG çıkışını, LMS ve NLMS algoritmalarının AGG çıkışları ile karşılaştırdığımızda FG algoritmasının, gürültüyü gidermede daha başarılı olduğunu söyleyebiliriz. Şekil 4.26, Şekil 4.27, Şekil 4.28, Şekil 4.29 ve Şekil 4.30 karşılaştırmalı olarak incelenilecek olursa; adaptif süzgecin derecesi arttığında, önerilen algoritma, gürültüyü gidermede başarılı olmaktadır. Yani incelediğimiz üç algoritma içerisinden, gürültüyü en başarılı şekilde önerilen FG algoritması bastırmıştır.

Tablo 4.15'te farklı süzgeç derecelerinde FG, LMS ve NLMS algoritmalarının SNR değerleri verilmektedir. Bu tablodan da görüleceği gibi gürültü gidericinin süzgeç derecesinin dört ile sekiz ve altı ile on olduğu durumlarda sırasıyla -3.6286 dB ile -3.8722 dB ve -4.4993 dB ile -4.6248 dB gibi birbirine yakın SNR değerlerinde çalışmasının başarımı gösterilirken, genel olarak iki işaretin toplamı olan bir bilgi işaretinde farklı süzgeç dereceleri ve farklı SNR değerlerinde adaptif gürültü gidericinin iyi bir başarım gösterdiği yukarıdaki şekillerden anlaşılmaktadır.

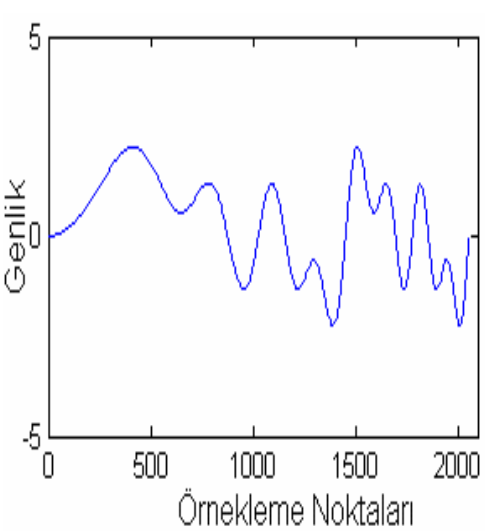
Tablo 4.15. $\sin(2\pi t^2/8N^2)+ \sin(2\pi t^2/4N^2)$ bilgi işareti için farklı süzgeç derecelerinde SNR değerleri.

Süzgeç derecesi	SNR(dB)
2	-2.2091
4	-3.6286
6	-4.4993
8	-3.8722
10	-4.6248

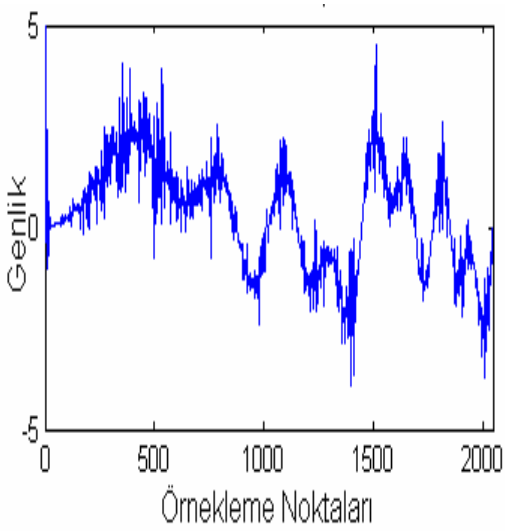
4.3.3. $\sin(2\pi t^2/8N^2)+\sin(2\pi t^2/4N^2)+\sin(2\pi t^2/2N^2)$ Bilgi İşareti Kullanılarak Farklı Süzgeç Dereceleri ile AGG Tasarımı:

Son olarak, farklı üç işaretin toplamından oluşan bilgi işareti için, farklı süzgeç dereceleri kullanıldığı durumda, AGG'nin başarımını ölçmek hedeflenmiştir. Bu alt bölüm içerisinde de, bilgi işaretinin, üç işaretin toplamı olarak ele alındığı durumda, yine önceki alt bölümlerde incelenilen, iki farklı bilgi işaretinde olduğu gibi, farklı süzgeç dereceleri için AGG'nin çıkışları incelenilerek karşılaştırılacaktır.

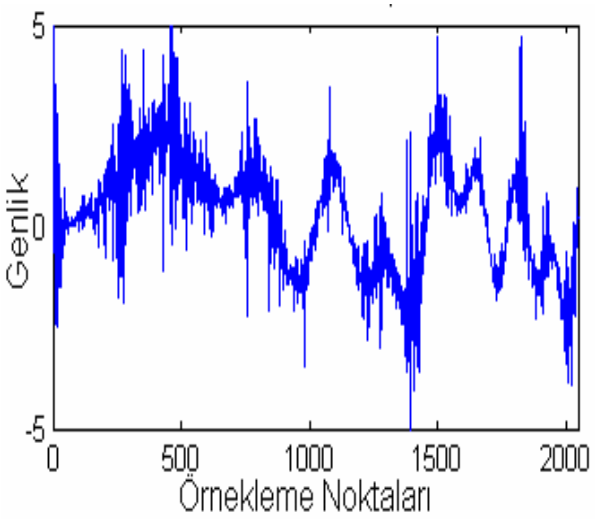
4.3.3.1. İkinci Dereceden Süzgeç Kullanılarak AGG:



(a)



(b)



(c)

Şekil 4.31. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi iki olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Bu bölümde Tablo 4.1'de verilen LMS ve NLMS algoritması için kullanılan kontrol parametreleri ve Tablo 4.3'de verilen FG algoritmasının kontrol parametreleri incelediğimiz üç işaret için kullanılmaktadır.

Şekil 4.31'de süzgeç derecesi iki olduğu zaman FG, LMS ve NLMS algoritmasının AGG çıkışı ve Tablo 4.16'da ise bu çalışmada elde edilen süzgeç katsayıları verilmektedir.

Tablo 4.16. FG, LMS ve NLMS algoritmaları ile ikinci dereceden süzgeç için elde edilen katsayılar.

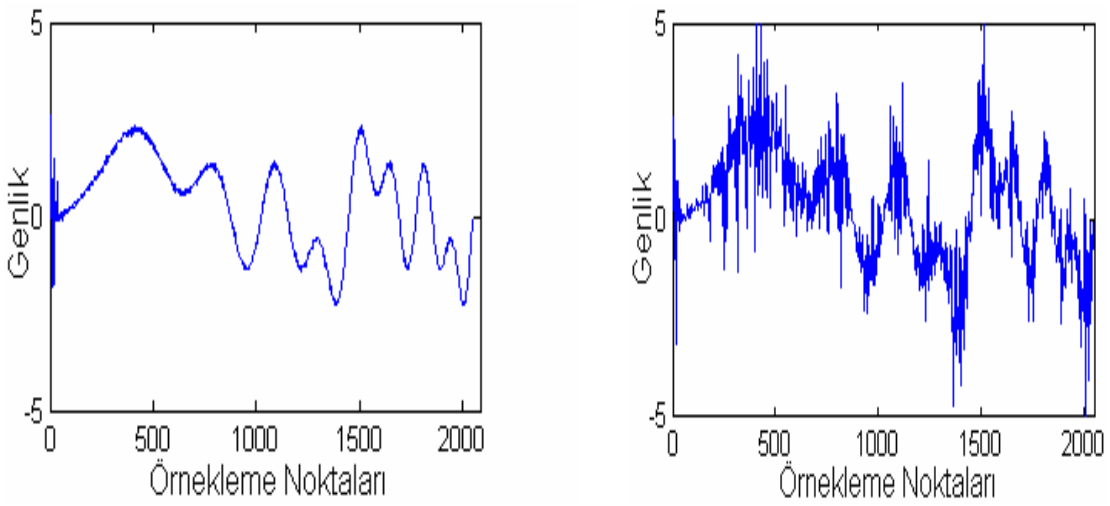
Katsayılar	**FG**	**LMS**	**NLMS**
W1	-1.2382	-1.2390	-1.2443
W2	-0.0362	-0.0589	0.0622

Şekil 4.31'de süzgeç derecesi iki olduğu zaman elde edilen FG algoritmasının AGG çıkışını, diğer iki klasik algoritmanın AGG çıkışı ile karşılaştırdığımızda, FG algoritmasının başarılı bir şekilde gürültüyü bastırdığı ve bilgi işaretinin yeniden elde

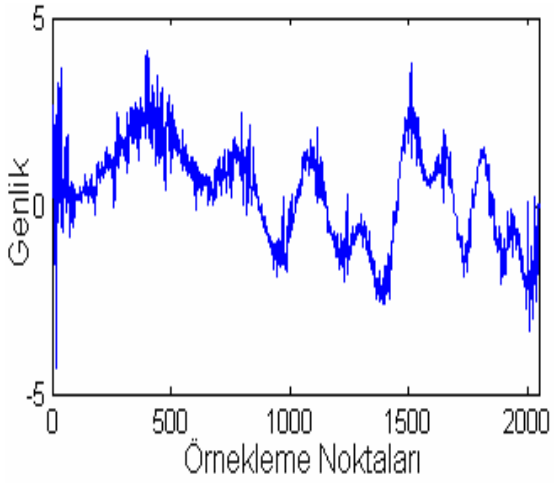
edilebildiği görülmektedir.

4.3.3.2. Dördüncü Dereceden Süzgeç Kullanılarak AGG:

Bu kısımda, süzgeç derecesinin dört olduğu durumda AGG'inin FG, LMS ve NLMS algoritmaları için gösterdikleri başarımlar incelenmiştir.



(a) (b)



(c)

Şekil 4.32. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi dört olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Şekil 4.32'de süzgeç derecesi dört olduğu zaman FG, LMS ve NLMS algoritmalarının

AGG çıkışı ve Tablo 4.17'de ise bu çalışmada elde edilen süzgeç katsayıları gösterilmektedir. Dördüncü dereceden süzgeç kullanıldığı durumda da, önerilen FG algoritması, başarılı bir şekilde gürültüyü bastırabilmektedir.

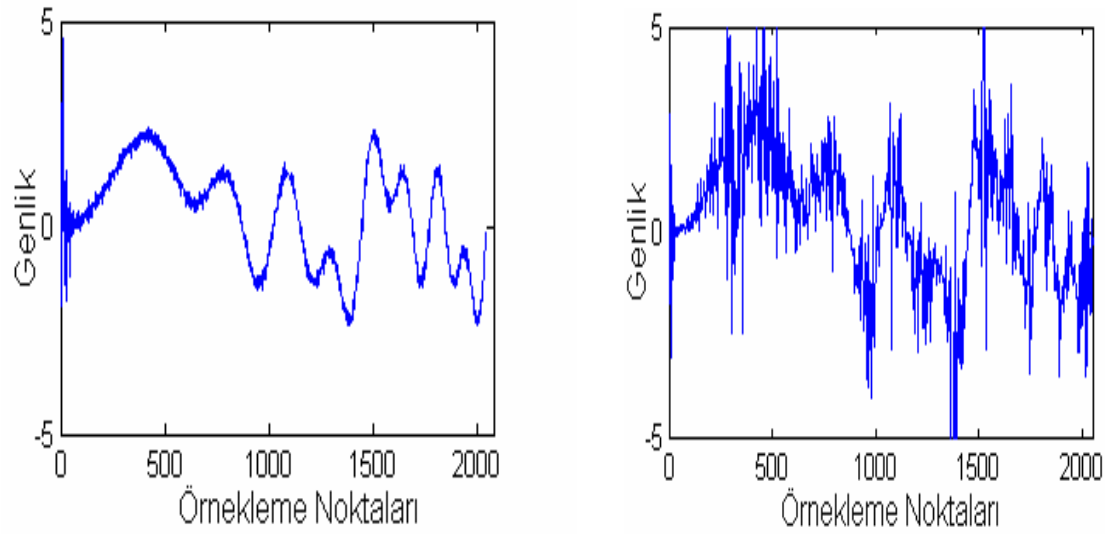
Tablo 4.17. FG, LMS ve NLMS algoritmaları ile dördüncü dereceden süzgeç için elde edilen katsayılar.

Katsayılar	FG	LMS	NLMS
W1	-0.9102	-0.8745	-0.7301
W2	-0.6808	-0.5931	-0.5130
W3	0.1018	0.1924	0.2187
W4	0.0201	0.0763	0.0373

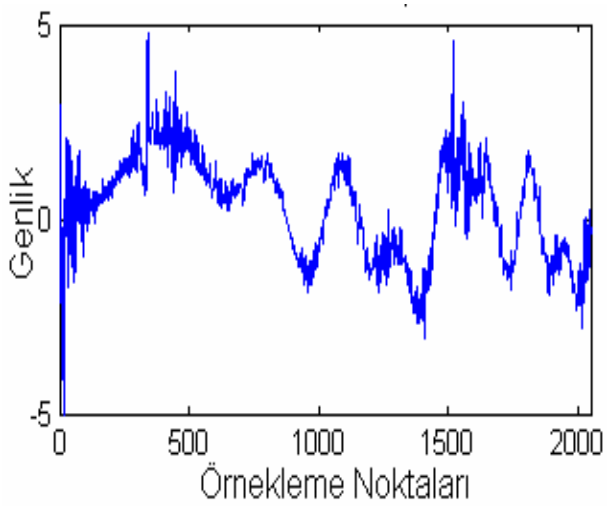
.

4.3.3.3. Altıncı Dereceden Süzgeç Kullanılarak AGG:

Bu bölümde, altıncı dereceden süzgeç kullanıldığı durumda FG, LMS ve NLMS algoritmalarının AGG çıkışı karşılaştırılmıştır. Şekil 4.33'de bu algoritmaların AGG çıkışları gösterilmektedir.



(a) (b)



(c)

Şekil 4.33. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi altı olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Tablo 4.18'de bu çalışmada elde edilen süzgeç katsayıları verilmektedir. Altıncı dereceden süzgeç kullanıldığı durumda da, önerilen FG algoritması, başarılı bir şekilde gürültüyü gidermektedir.

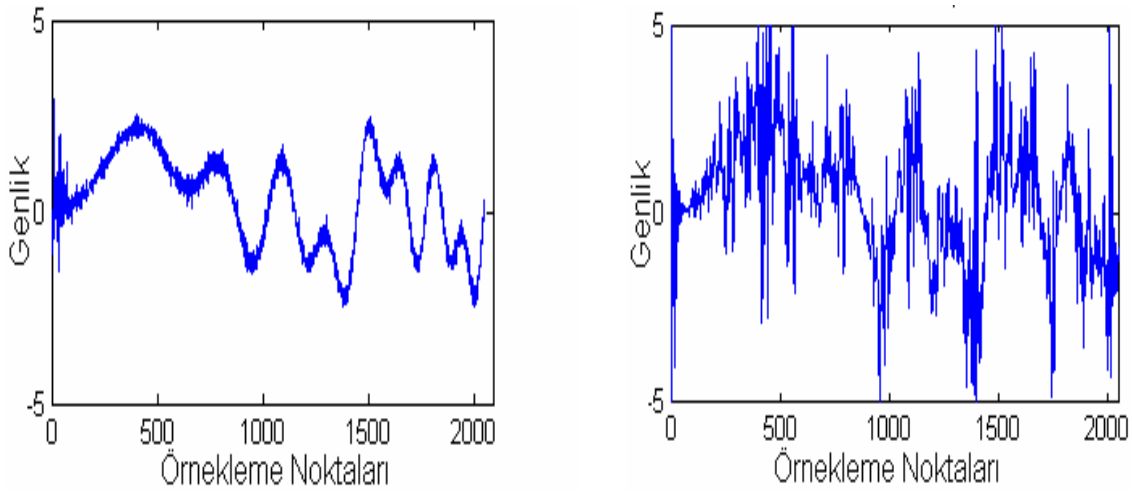
Tablo 4.18. FG, LMS ve NLMS algoritmaları ile altıncı dereceden süzgeç için elde edilen katsayılar.

Katsayılar	**FG**	**LMS**	**NLMS**
W1	-0.4744	-0.5219	-0.2312
W2	0.2530	0.2467	0.3222
W3	0.7248	0.7105	0.7125
W4	0.2368	0.2246	0.3350
W5	0.6527	0.5995	0.6494
W6	-0.5616	-0.6461	-0.3625

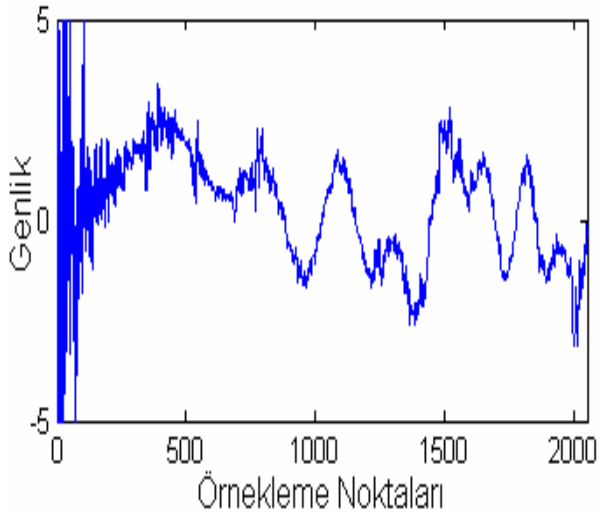
4.3.3.4. Sekiz Dereceden Süzgeç Kullanılarak AGG:

Süzgeç derecesi sekiz olduğu zaman AGG çıkışları bu bölümde incelenmiş ve Şekil

4.34'de AGG çıkışları gösterilmiştir.



(a) (b)



(c)

Şekil 4.34. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi sekiz olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

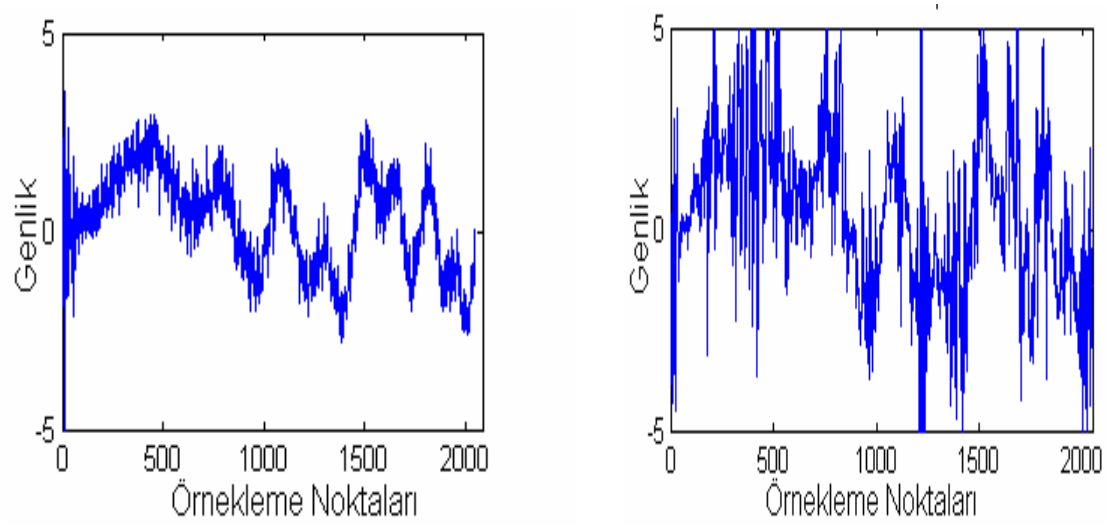
Tablo 4.19'da ise FG, LMS ve NLMS algoritmalarında sekizinci dereceden süzgeç kullanıldığında elde edilen süzgeç katsayıları verilmektedir. Yine burada da, Şekil 4.34'de görüldüğü gibi, önerilen FG algoritmasının gürültüyü gidermede, incelediğimiz diğer iki algoritmaya göre daha iyi bir başarı sergilediği görülmüştür.

Tablo 4.19. FG, LMS ve NLMS algoritmaları ile sekizinci dereceden süzgeç için elde edilen katsayılar.

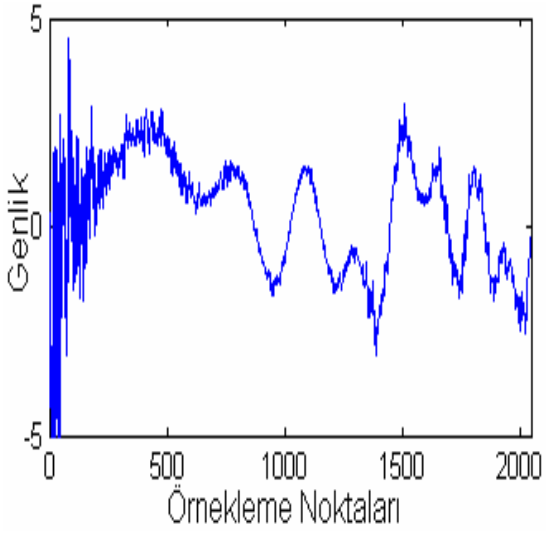
Katsayılar	FG	LMS	NLMS
W1	0.0847	-0.1473	0.1129
W2	0.5265	0.3855	0.7222
W3	-0.5621	-0.6057	-0.1562
W4	-0.2470	-0.4241	-0.1404
W5	2.9150	3.2988	2.6797
W6	-0.8729	-1.1828	-0.6216
W7	-0.8749	-1.0204	-0.6139
W8	-0.2156	-0.1789	0.2252

4.3.3.5. Onuncu Dereceden Süzgeç Kullanılarak AGG:

Onuncu dereceden süzgeç kullanıldığı durumda da, FG, LMS ve NLMS algoritmalarının gürültü giderici çıkışları bu alt bölümde incelenmiştir. Şekil 4.35'de bu algoritmaların AGG çıkışları gösterilmektedir.



(a) (b)



(c)

Şekil 4.35. Üç sinüzoidal işaretin toplamı olan bilgi işaretinde ve süzgeç derecesi on olduğunda (a) FG algoritmasının (b) LMS algoritmasının (c) NLMS algoritmasının AGG çıkışı.

Tablo 4.20. FG, LMS ve NLMS algoritmaları ile onuncu dereceden süzgeç için elde edilen katsayılar.

Katsayılar	**FG**	**LMS**	**NLMS**
W1	-0.8327	-0.8157	-0.1827
W2	-0.5758	-0.7176	0.0664
W3	-0.0632	-0.2755	0.2838
W4	-0.3427	-0.3258	0.0672
W5	0.1753	-0.0644	0.2882
W6	0.8313	1.0921	1.0102
W7	-0.5030	-0.4896	-0.0337
W8	0.3621	-0.0445	0.0978
W9	0.4756	1.1151	0.8702
W10	-0.7808	-0.9978	-0.5671

Tablo 4.20'de ise FG, LMS ve NLMS algoritmalarında onuncu dereceden süzgeç için elde edilen katsayılar verilmektedir. Süzgeç derecesi on olduğu ve üç farklı sinüzoidal

işaretin toplamı olan bir bilgi işareti kullanıldığı zaman da, önerilen FG algoritmasının AGG çıkışı ile LMS ve NLMS algoritmalarının AGG çıkışı karşılaştırıldığında yine FG algoritmasının gürültüyü gidermede daha hızlı ve daha başarılı olduğu söylenebilmektedir.

Şekil 4.31, Şekil 4.32, Şekil 4.33, Şekil 4.34 ve Şekil 4.35 incelenecek olursa, farklı olarak alınan bütün süzgeç derecelerinde, FG algoritması, gürültüyü gidermede başarılı olmaktadır. Bu üç algoritma içerisinden, gürültüyü bastırmada yine en başarılı olan algoritma, önerdiğimiz FG algoritması olmuştur.

Tablo 4.21'de farklı süzgeç derecelerinde FG, LMS ve NLMS algoritmalarının SNR değerleri verilmektedir. Bu bölümdeki üç farklı işaretin toplamı olarak alınan bilgi işaretinin gürültü giderici cevabını incelediğimizde, farklı süzgeç dereceleri için yakın SNR değerindeki gürültüyü başarılı bir şekilde giderdiği görülürken aynı süzgeç derecelerinde farklı SNR değerlerindeki gürültü içinde benzer sonuçlar elde edilmiştir.

Tablo 4.21. $\sin(2\pi t^2/8N^2)+ \sin(2\pi t^2/4N^2)+ \sin(2\pi t^2/2N^2)$ bilgi işareti için farklı süzgeç derecelerinde SNR değerleri.

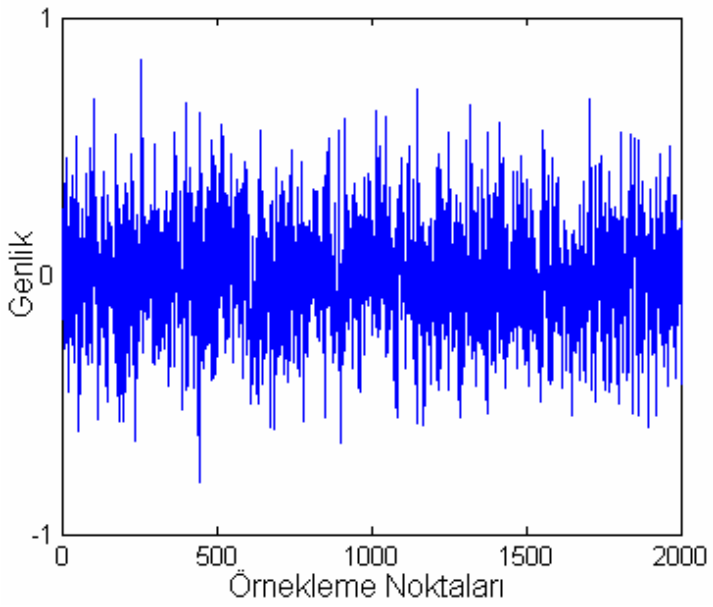
Süzgeç derecesi	**SNR(dB)**
2	-0.0117
4	0.2492
6	-0.4750
8	-4.8394
10	-2.5372

Çalışmayla ilgili LMS ve NLMS algoritmaları ve FG algoritmasının temel adımları Ek-1'de verilmektedir.

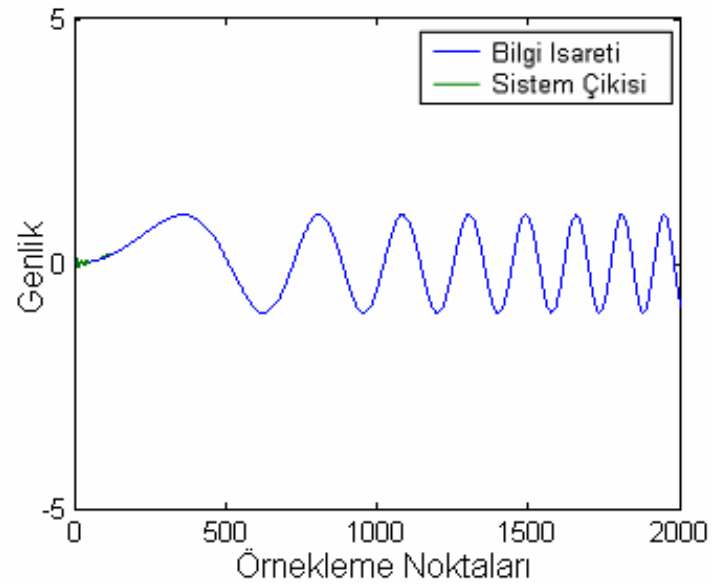
4.4. FARKLI SNR DEĞERLERİNDE AGG'NİN BAŞARIMI

Bu bölümde, AGG için yazılan FG algoritmasının başarımı, farklı SNR değerleri için

incelenmiştir. Bilgi işareti olarak; $\sin(2\pi t^2/8N^2)$ işareti, işaret gürültü oranı olarak ise 10 dB ve 20 dB'lik iki SNR değeri alınmış ve ikinci dereceden bir süzgeç üzerinde incelemeler yapılmıştır. Şekil 4.3'de AGG girişi tek bir sinüzoidal işaretten oluşan s(n) bilgi işareti, Şekil 4.36'da 10 dB değerindeki $n_0(n)$ gürültü işareti gösterilmektedir. AGG tasarımda kullanılan FG algoritmasının kontrol parametreleri, önceki çalışmalarda kullanılan değerlerle aynı olarak, Tablo 4.3'deki gibi alınmıştır.

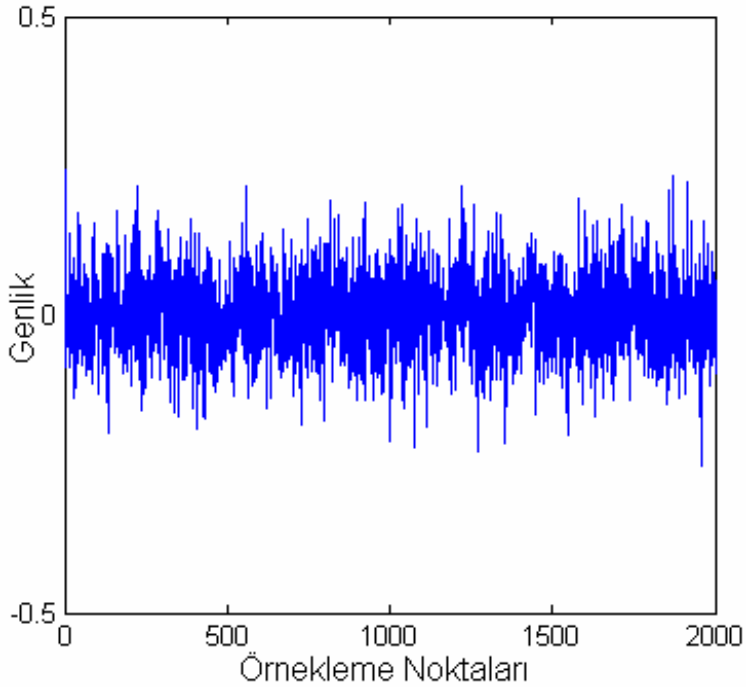


Şekil 4.36. 10 dB değerindeki gürültü işareti, $n_0(n)$.

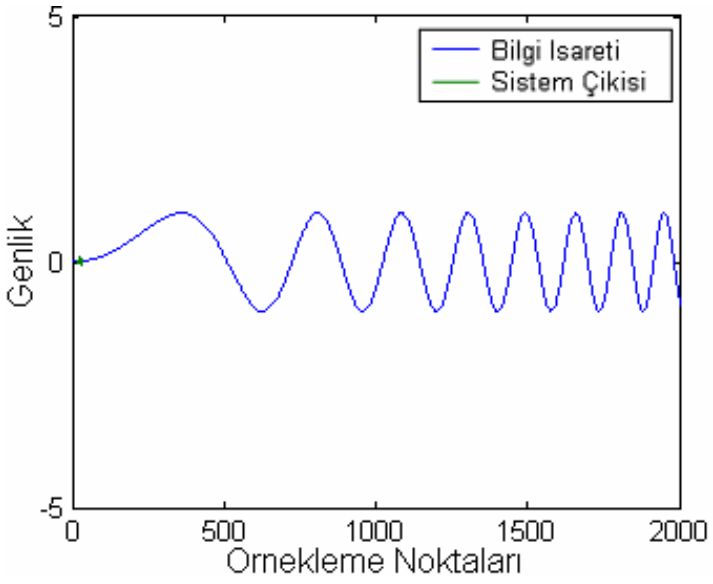


Şekil 4.37. SNR değeri 10 dB olduğu zaman AGG çıkışı.

Şekil 4.37'de SNR değeri 10 dB olduğu zaman FG algoritması kullanıldığında, AGG'nin çıkışı verilmiştir. Şekil 4.38'de 20 dB değerindeki $n_0(n)$ gürültü işareti gösterilmektedir. Şekil 4.39'da ise SNR değeri 20 dB olduğunda FG algoritması ile elde edilen AGG çıkışı verilmektedir.



Şekil 4.38. 20 dB değerindeki gürültü işareti, $n_0(n)$.



Şekil 4.39. 20 dB değerindeki gürültü işaret, $n_0(n)$.

Şekil 4.37 ve Şekil 4.39'dan da anlaşılmaktadır ki süzmede kullanılan hata ε, bu çalışma ile elde edilmek istenen işareti vermektedir. Artan SNR değerlerinde gürültü giderici istenildiği gibi gürültüyü çabuk ve başarılı bir şekilde çalışarak gidermektedir. SNR değerleri 10 ve 20 dB olduğunda, algoritma performansını değerlendirmek amacıyla elde edilen en iyi sonuç, şekillerin çizdirilmesinde kullanılmıştır. Adaptasyon algoritması olarak FG algoritması tercih edilirse, değişik SNR değerlerinde de başarılı bir adaptif gürültü giderici tasarlanabildiği simülasyon sonuçlarından görülmüştür. SNR değerlerini arttırdığımızda gürültü gidericinin daha kısa zamanda gürültüyü giderdiği gösterilmiştir [76].

4.5. FG ALGORİTMASININ KONTROL PARAMETRELERİNİN AGG TASARIMINA ETKİSİ

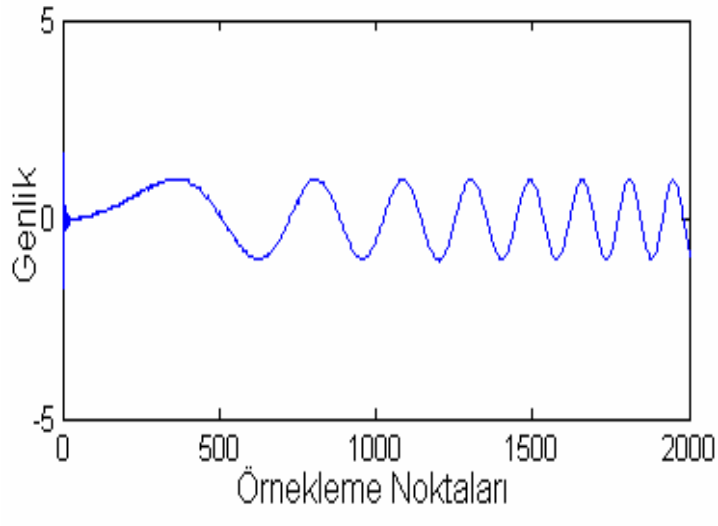
Bu bölümde FG algoritmasının farklı kontrol parametre değerlerininin AGG tasarımına etkisi araştırılmıştır. Bunun için birleşme faktörü ve ölçekleme faktörü temel alınarak AGG tasarımı gerçekleştirilmiştir.

Bilgi işareti olarak, Bölüm 4.4'de incelediğimiz $\sin(2\pi t^2/8N^2)$ işareti ve ikinci dereceden süzgeç kullanılmıştır. Şekil 4.3'de AGG girişi tek bir sinüzoidal işaretten oluşan s(n) bilgi işareti gösterilmiştir. Birinci aşamada, birleşme faktörü K'nın değerinin 0.4 ve 0.6 olduğu durumda AGG tasarımına etkisi incelenmiştir. K'nın değeri 0.4 alındığında FG algoritmasının kullanılan diğer parametreleri, Tablo 4.22'de verilmiştir.

Tablo 4.22. FG algoritmasında kullanılan diğer kontrol parametreleri.

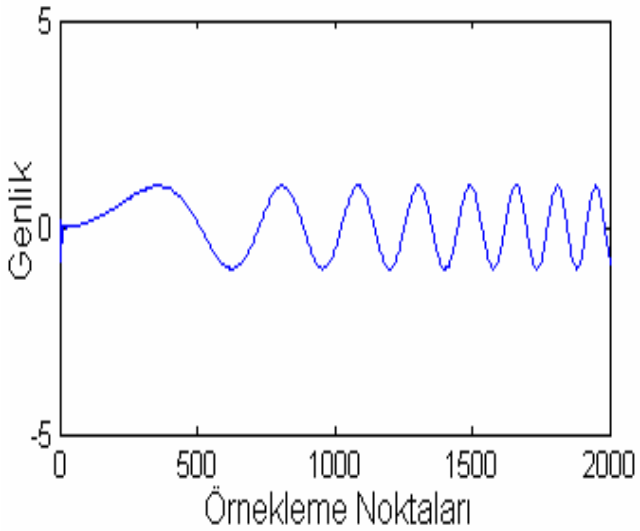
Popülasyon büyüklüğü(NP)	100
Çaprazlama oranı (CR)	0,8
Ölçekleme faktörü (F)	0,5
İterasyon sayısı (Gmax)	2048

Şekil 4.40'da FG algoritması kullanıldığında K değeri 0.4 iken AGG'nin çıkışı verilmektedir.



Şekil 4.40. K değeri 0.4 iken AGG çıkışı.

K için 0.6 değeri alınırken, Tablo 4.22'deki FG'nin diğer kontrol parametreleri bu tasarımda da geçerli olmaktadır. Şekil 4.41'de FG algoritmasının K değeri 0.6 iken elde edilen AGG'nin çıkışı verilmektedir.



Şekil 4.41. K değeri 0.6 iken AGG çıkışı.

Bu çalışmada, adaptif gürültü giderici tasarımında kullanılan FG algoritmasının kontrol parametrelerinden olan birleştirme ve ölçekleme faktörünün seçiminde, algoritmayı hızlandırmak için değişiklikler yapılmıştır [77]. Bu iki parametre için bu çalışmada 0,4 ile 0.6 olmak üzere iki değer alınmıştır. Şekil 4.40 ve Şekil 4.41'de görüldüğü gibi

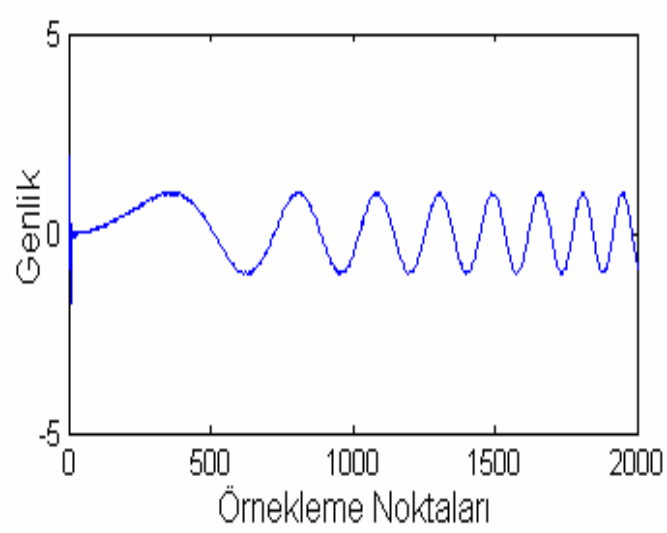
K'nın her iki değerinde de AGG çıkışımız siüzoidal s(n) bilgi işaretimizi yeniden elde etmeyi başarmıştır.

İkinci aşamada ise, ölçekleme faktörü F'nin değerinin 0.5 ve 0.8 olduğu durumda AGG tasarımına etkisi incelenmiştir. F'nin değeri 0.5 alındığında FG algoritmasının kullanılan diğer parametreleri Tablo 4.23'de verilmiştir.

Tablo 4.23. FG algoritmasında kullanılan diğer kontrol parametreleri.

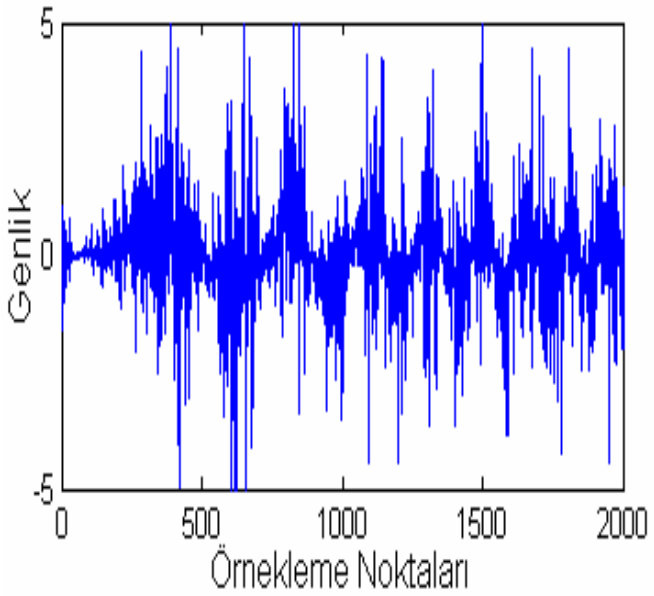
Popülasyon büyüklüğü(NP)	100
Çaprazlama oranı (CR)	0,8
Birleşme faktörü (K)	0,4
İterasyon sayısı (Gmax)	2048

Şekil 4.42'de FG algoritması kullanıldığında F değeri 0.5 iken AGG'nin çıkışını göstermektedir.



Şekil 4.42. F değeri 0.5 iken AGG çıkışı.

Şekil 4.43'de, F değeri 0.8 ve kullanılan diğer FG parametreleride Tablo 4.23'deki gibi iken AGG'nin çıkışını göstermektedir.



Şekil 4.42. F değeri 0.8 iken AGG çıkışı.

Şekil 4.42'den görüldüğü gibi F değeri 0.4 olduğunda istenilen bilgi işareti elde edilebilirken, F değeri 0.8 olduğunda gürültü bastırılamamakta ve elde etmek istediğimiz bilgi işareti elde edilememektedir.

5. BÖLÜM

SONUÇ VE ÖNERİLER

Gürültü giderme, birçok mühendislik dalında önemli bir problem olarak karşımıza çıktığından, haberleşme sistemlerinde, bilgi işaretinin gürültüden arındırılması gerekmektedir. Gürültü giderme, gürültüyle bozulmuş bilgi işaretinin, gürültü bileşenlerinden arındırılarak yeniden elde edilmesidir. Gerçekleştirilen tez çalışmasında, adaptif FIR süzgeçlerde, gürültü giderme konusu incelenerek bunun için bir yapay zeka optimizasyon algoritması olan FG algoritmasına dayalı bir AGG yapısı önerilmiştir. Yazılım bütün programlama dillerine uyarlanabilecek basitlikte yapılmıştır.

Gerçekleştirilen tez çalışmasında öncelikle, adaptif FIR süzgeçler için, literatürde oldukça sık kullanılan klasik tekniklerden olan LMS ve NLMS algoritmalarıyla AGG tasarlanmıştır. Daha sonra gürültü giderme için bir önerilen FG algoritması AGG tasarımında kullanılarak LMS ve NLMS algoritmaları ile elde edilen sonuçlarla karşılaştırılmıştır. FG algoritması kullanıldığında gürültünün daha iyi bastırıldığı yapılan simülasyon çalışmalarından belirlenmiştir. Bu çalışmada bütün simülasyon uygulamaları için hiçbir bağımlı değişkenle ilişkili olmayan ve beyaz gürültü olarak da adlandırılan ideal gürültü çeşidi olan ve literatürde çokça kullanılan rastgele gürültü kullanılmıştır. Algoritmanın başarımını ölçmek için, üç farklı sinüzoidal işaret temel alınarak AGG'nin çıkışları incelenmiştir. İlk işaret tek bir sinüs bileşeninden, ikinci işaret iki farklı sinüs işaretinin toplamından ve üçüncü işarette üç farklı sinüs işaretinin toplamından oluşacak şekilde alınmıştır. Bu üç farklı bilgi işareti içinde ikinci, dördüncü, altıncı, sekizinci ve onuncu dereceden adaptif FIR süzgeçler kullanılarak, LMS, NLMS ve FG algoritmalarının başarımları incelenmiştir. LMS NLMS ve FG algoritmaları kullanılarak beş farlı süzgeç derecesi için AGG'nin çıkışları incelendiğinde önerilen FG algoritmasının, LMS ve NLMS algoritmalarına göre daha

hızlı ve çok daha hassas sonuçlar verdiği gözlemlenmiştir.

Ayıca bu çalışmalara ek olarak farklı işaret/gürültü oranlarına sahip rastgele gürültü işaretiyle farklı süzgeç derecelerinde ve farklı bilgi işaretlerinde, AGG'nin çıkışı incelenmiş ve elde edilen sonuçlar değerlendirilmiştir. Farklı süzgeç dereceleri ile, üç farklı sinüs bilgi işareti kullanılarak, bilgi işaretinin rastgele gürültü işareti ile bozulduğu durumlar üzerinde çalışıldığında, önerilen FG algoritması kullanıldığı durumda adaptif süzgecin, gürültü gidermedeki başarımı incelenmiş ve kullanılan adaptif süzgecin derecesi arttıkça gürültü gidericinin başarımında düşüş gözlemlenmesine karşın, LMS ve NLMS ile elde edilen AGG çıkış değerleriyle karşılaştırıldığında gürültüyü iyi bir şekilde engellediği gözlemlenmiştir. Kullanılan adaptif süzgecin derecesi büyüdükçe LMS ve NLMS algoritmaları ile elde edilen gürültü gidericinin başarımlarının düştüğü ve arzu edilen gürültüyü bastırıp bilgi işaretini yeniden elde etmekte zorlandığı görülmüştür. Bu çalışma için önerdiğimiz FG algoritması ise, daha büyük süzgeç derecelerine sahip AGG'lerde, gürültüyü bastırıp istenen bilgi işaretini elde etmede daha başarılı olmuştur. Farklı bilgi işaretlerinde de gürültü gidericinin aynı başarımda çalışarak gürültü giderdiği görülmüştür. Daha sonra 10 ve 20 dB değerlerinde iki farklı SNR değerinde, FG algoritmasının başarımı değerlendirilmiş ve başarılı bir şekilde gürültünün giderildiği görülmüştür. Diğer bir karşılaştırma ise FG algoritmasının kontrol parametrelerinden olan birleşme faktörü ve ölçekleme faktörünün iki farklı değerinde AGG'nin başarımı incelenmiştir. Birleşme faktörü 0.4 ve 0.6 gibi iki farklı değer alınarak her iki değerde de iyi bir başarım elde edildiği görülmüştür. Ölçekleme faktörü değerleri ise 0.5 ve 0.8 olarak alınmıştır. F değeri 0.5 iken gürültü giderilebilirken F değerinin 0.8 olarak arttırıldığında başarımda bir düşüş gözlenmiş ve elde edilmek istenen bilgi işareti AGG çıkışımızdan elde edilememiştir.

Sonuç olarak, yapay zeka algoritmalarından biri olan FG algoritması, gürültü giderme için kullanılabileceği gibi, kimliklendirme, modelleme ve çeşitli adaptif süzgeç yapılarının tasarlanması gibi uygulamalarda kullanılabilir. Aynı zamanda, farklı yapay zeka optimizasyon algoritmaları kullanılarak da bu tez çalışmasında uygulanan tasarımlar gerçekleştirilebileceği gibi tasarımların donanım bazında gerçekleştirilmesi de mümkündür.

KAYNAKLAR

1. Koyuncu, C.A., Farksal Gelişim Algoritmasının İncelenmesi ve İşaret Kestiriminde Kullanılması, Erciyes Üniversitesi, Yüksek Lisans Tezi, Kayseri, 2006.

2. Price, K., Storn, R., A Simple Evolution Strategy for Fast Optimization. Dr. Dobb's Journal, 264, 18-24, 1997.

3. Price, K, Storn, R, http://www.ICSI.Berkeley.edu/~storn/code.html.

4. Storn, R., Differential Evolution Design of an IIR-Filter with Requirements for Magnitude and Group Delay. In: Proceedings IEEE International Conference on Evolutionary Computation ICEC'96. IEEE Press, New York, 268 – 273, 1996.

5. Comon, P., Pham, D. T., Estimating The Order of A FIR Filter for Noise Cancellation, IEEE Trans. On Information Theory, 36, 2, 1990.

6. Graupe, D., Efron, J. A., An Output-Whitening Approach to Adaptive Active Noise Cancellation, IEEE Trans. On Circuits and Systems, 38, 11, 1306–1313, 1991.

7. Kwan, H. K., Li, Q. P., New Nonlinear Adaptive FIR Digital Filter For Broadband Noise Cancellation, IEEE Trans. On Circuits and Systems-II. Analog and Digital Signal Processing, 41, 5, 355–360, 1994.

8. Fah, L. B., Hussain, A., Samad, S. A., Speech Enhancement by Noise Cancellation Using Neural Network, IEEE, 2000.

9. Kang, G. S., Fransen, L. J., Experimentation with an Adaptive Noise-Cancellation Filter, IEEE Transaction On Circuits and Systems, 34, 7, 753-758, 1987.

10. Savoji, M. H., A Variable Lenght Lattice Filter For Adaptive Noise Cancellation, ICASSP 86, 2935-2938, Tokyo.

11. Li, Z., Er, M. J., Adaptive Noise Cancellation Using IIR- Based Fuzzy Systems, IEEE International Conference on Systems, 5691-5696.

12. Xie, Q., Kwan, K., Multichannel Lattice Structure For Adaptive Noise Cancellation, Proceeding of 2004 International Symposium on Intelligent Multimedia, Video and Speech Proceeding, Hong Kong, 326-329, October, 2004.

13. Li, Z., Er, M. J., Cai, H., Chen, Q., Adaptive Noise Cancellation Using Enhanced Dynamic Fuzzy Neural Systems, IEEE Transaction On Fuzzy Systems, 13, 3, 391-342, 2005.

14. Li, C., Cheng, K. H., Chen,C. M., Chen, J. L., Soft Computing Approach to Adaptive Noise Filtering, Proceeding of 2004 IEEE Conference on Cybernetics and Intelligent Systems Singapore, December, 2004.

15. Kao, W. C., Hung, Y. W., Hsu, C. C, Kao, C. C., Chen, S. H., Adaptive Fixed Pattern Audio Noise Cancellation for Digital Camera Applications, IEEE 2006.

16. Shimauchi, S., Haneda, Y., Kataoka, A., A Robust NLMS Algorithm for Acoustic Echo Cancellation, Electronics and Communications in Japan Part3, 89, 11, 2006.

17. Efron, A. J., Han, L. C., Wide-Area Adaptive Active Noise Cancellation, 41, 6, 405-409, 1994.

18. Mayer, D.G., Kinghorn, B.P., Archer A. A. Differential evolution – An Easy and Efficient Evolutionary Algorithm for Model Optimisation, Agricultural Systems, 83, 315-328, 2005.

19. Shiakolas, P. S., Koladiya, D., Kebrle, J., Optimum Robot Design Based on Task Specifications Using Evolutionary Techniques and Kinematic, Dynamic, and Structural Constraints. International Journal of Inverse Problems in Engineering, 10, 4, 359–375, 2002.

20. Karaboğa, N., Çetinkaya,B., Efficient Design of Fixed Point Digital FIR Filters by Using Differential Evolution Algorithm. Lecture Notes in Computer Science, 3512, 2005.

21. Kukkonen, S. Lampinen, J., An Emprical Study of Control Parameters for

Generalized Differential Evolution, International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (Eurogen 2005), Germany, Sept 12-14, 2005.

22. Rönkkönen, J., Kukkonen, S., Price, K., Real-Parameter Optimization with Differential Evolution, IEEE Congress on Evolutionary Computation, Edinburg, Scotland, 1, 506–513, September 2-5, 2005.

23. Kukkonen, S., Lampinen, J., GDE3:The third Evolution Step of Generalized Differential Evolution, IEEE Congress on Evolutionary Computation (CEC 2005), Edinburgh, United Kingdom, 443-450, Sept 2-5, 2005.

24. Price, K., Storn, R., Lampinen, J., Differential Evolution: A Practical Approach to Global Optimization, Springer-Verlag, Germany, 538, 2005.

25. Zielinski, K., Peters, D., Laur, R., Constrained Multi-Objective Optimization Using Differential Evolution, Proceedings of the Third Initernational Conference on Computational Intelligence, Robotics and Autonomuous Systems, Singapore, 2005.

26. Zielinski, K., Peters, D., Laur, R., Run Time Analysis Regarding Stopping Criteria for Differential Evolution and Particle Swarm Optimization, Proceedings of the 1st International Conference on Experiments/Process/System Modelling/Simulation/Optimization, Athens, 2005.

27. Weitkemper, P., Zielinski, K., Kammeyer, K. D., Laur, R., Optimization of Interference Cancellation in Coded CDMA Systems by means of Differential Evolution, 2005.

28. O'neill, M., Brabazon, A., Grammatical Differential Evolution, 2006 International Conference on Artificial Intelligence (ICAI'06) CSEA Press Las Vegas, Nevada, 2006.

29. Robic T., Filipic B., DEMO: Differential Evolution for Multiobjective Optimization, Evolutionary Multi-Criterion Optimization Lecture Notes In

Computer Science, 3410, 520-533, 2005.

30. Braak, C. J. F., Ter, A. Marcov Chain Monte Carlo Version of the Genetic Algorithm Differential Evolution: Easy Bayesian Computing for Real Parameter Spaces. Statistics and Computing 16, 3, 1573–1375, 2006.

31. Besson, P., Vesin, J., Popovici, V., Kunt, M., Differential Evolution Applied to a Multimodal Information Theoretic Optimization Problem, Lecture Notes in Computer Science, 3907–2006, 505–509, 2006.

32. Karaboğa, N., Çetinkaya,B., Noise Cancellation by Using Genetic Algorithm Based Adaptive FIR Filters, Sinyal İşleme ve İletişim Uygulamaları Kurultayı SİU 2007, 2007.

33. Widrow, B. Stearns, S. D., Adaptive Signal Processing, 302-306, Prentice Hall, 1986.

34. Haykin, S., Adaptive Filter Theory, Prentice-Hall, 2002.

35. Sankaran, Sundar G., On Ways to Improve Adaptive Filter Performance, Ph.D Thesis, Virginia Tech, India, 1999.

36. Jiang, D., A Study on Adaptive Filtering for Noise and Echo Cancellation, Canada, 2005.

37. Reglia, P. A., Stable and Efficient Lattice Algorithms for Adaptive IIR Filtering, IEEE Trans. Signal Procesing, 40, 2, 375–388, 1998.

38. Lopez-Valcarce, R., Perez-Gonzalez, F., Adaptive Lattice IIR Filtering Revisited: Convergence Issues and New Algorithms with Improved Stability Properties, IEEE Trans. Signal Procesing, 49, 4, 811–821, 2001.

39. Rao, B. D., Adaptive IIR Filtering using Cascade Structure, IEEE, 194–198, 1993.

40. Shynk, J. J., Adaptive IIR Filtering using Parallel Form Realizations, IEEE Trans. Acoust. Speech, Signal Procesing, 37, 4, 519–533, 19.

41. Zeng, G., Adaptive Digital Filtering, Ph.D Thesis, Arizona State University,

USA, 1990.

42. Sastry, K. K. N., Behera, L., Nagrath, I. J., A Self Organizing Fuzzy Controller Design Using Differantial Evoluation, In Sixth Symposium on Intelligent Systems, IISc Bangalore, 166-177, 1997.

43. Bose, T., Venkatachalam, A., Thamvichai, R., Multiplierless Adaptive Filtering, Digital Signal Processing, 12, 107-118, 2002.

44. Widrow, B., Adaptive Noise Cancelling: Principles and Applications, Proceddings IEEE, 63, 1692-1716, 1975.

45. Iliev, G., Kasabov, N., Adaptive Blind Noise Suppression in Some Speech Processing Applications, 2.

46. Yang, G., Chen, I-Ming: Task Based Optimization of Modular Robot Configurations: Minimized Degree-of-Freedom Approach, Mechanism and Machine Theory, 35, 517–540, 2000.

47. Han, J., Chung, W. K., Youm, Y., Kim, S. H., Task Based Design of Modular Robot Manipulator using Efficient Genetic Algorithm. Proceedings of IEEE International Conference on Robotics and Automation, 1, 507–512, 1997.

48. Parker, J. K., Khoogar, A. R., Goldberg, D. E., Inverse Kinematics of Redundant Robots Using Genetic Algorithms, Proceedings of IEEE International Conference on Robotics and Automation, 1, 271–276, 1989.

49. Yun, W. M., Xi, Y. G., Optimum Motion Planning in Joint Space For Robots Using Genetic Algorithm, Robotics and Autonomous Systems, 18(4), 373-393, 1996.

50. Storn, R., Price, K., Differential Evolution - A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces, Journal of Global Optimization, Kluwer Academic Publishers, 11, 341 – 359, 1997.

51. Price, K. V., Differential Evolution: A Fast and Simple Numerical Optimizer, In: Smith, M., Lee, M., Keller, J., (eds.): Biennial Conference of

the North American Fuzzy Information Processing Society, NAFIPS, IEEE Press, New York, 524-527, 1996.

52. Karaboğa, D., Yapay Zeka Optimizasyon Algoritmaları, Nobel Yayınevi, İstanbul, 2004.

53. Vondras, J., Different Evolution Algorithm Application in Optimized Filter Design, In Proceedings of Workshop 2001. Prague: CTU, A, 50–51, 2001.

54. Wang, F. S., Jing, C. H., Tsao, G. T., Fuzzy- Decision Making Problems of Fuel Ethanol Production Using Genetically Engineered Yeast, Industrial & Engineering Chemistry Research, 37, 3434–3443, 1998.

55. Sastry, K.K.N., Behera, L., Nagrath, I.J., Differential Evolution Based Fuzzy Logic Controller for Nonlinear Process Control, Fundamenta Informatica, 37, 121-136, 1999.

56. Wang, F. S., Cheng, W. M., Simultaneous Optimization of Feeding Rate and Operation Parameters for Fed-batch Fermentation Processes, Biotechnology Progress, 15(5), 949-952, 1999.

57. Chiou, J. P., Wang, F. S., Hybrid Method of Evolutionary Algorithms for Static and Dynamic Optimization Problems with Application to a Fed- Batch Fermentation Process, Computers & Chemical Engineering, 23, 1277-1291, 1999.

58. Joshi, R., Sanderson, A. C., Minimal Representation of Multi-Sensor Fusion Using Differential Evolution, IEEE Transactions on Systems, Man and Cybernetics, 29, 63-76, 1999.

59. Lee, M. H., Han, C., Chang, K. S., Dynamic Optimization of a Continuous Polymer Reactor Using a Modified Differential Evolution, Industrial & Engineering Chemistry Research, 38, 4825-4831, 1999.

60. Babu, B. V., Sastry, K. K. N., Estimation of Heat-Transfer Parameters in a Trickle-bed Reactor Using Differential Evolution and Orthogonal Collocation, Computers & Chemical Engineering, 23, 327-339, 1999.

61. Babu, B. V., Munawar, S. A., Differential Evolution for The Optimal Design of Heat Exchangers, Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, IE (I), India, March 11, 2000.

62. Babu, B. V., Munawar, S. A., Optimal Design of Shell & Tube Heat Exchanger by Different Strategies of Differential Evolution, PreJournal.org – The Faculty Lounge, 003873, http://www.prejournal.org, 2001.

63. Babu, B. V., Singh, R. P., Synthesis & Optimization of Heat Integrated Distillation Systems Using Differential Evolution, Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, IE (I), India, March 11, 2000.

64. Babu, B. V., Angira, R., Optimization of Nonlinear Functions Using Evolutionary Computation, Proceedings of 12th ISME Conference, India, 10−12, 153-157, January, 2001a.

65. Babu, B. V., Angira, R., Optimization of Thermal Cracker Operation Using Differential Evolution, Proceedings of 54th Annual Session of IIChE, Chennai, December 19-22, 2001b.

66. Ilonen, J., Kamarainen, J. -K., Lampinen, J. Differential Evolution Training Algorithm for Feed-Forward Neural Networks, In: Neural Processing Letters 7, 1, 93-105, 2003.

67. Rogalsky, T., Derksen, R.W., Kocabiyik, S., Differential Evolution in Aerodynamic Optimization, In: Proceedings of the 46th Annual Conference of the Canadian Aeronautics and Space Institute, 29-36, 1999.

68. Karaboga, D., Okdem, Ö., A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm. Turk J Elec Engin, 12, 1, 2004.

69. Karaboga,K, Digital IIR Filter Design Using Differential Evolution Algorithm. EURASIP Applied Signal Processing, 8, 1269-1276, 2005.

70. Babu, B. V., Angira, R., Optimization of Thermal Cracker Operation Using

Differential Evolution, Proceedings of International Symposium & 54th Annual Session of IIChE (CHEMCON-2001). Chennai, December 19-22, 2001.

71. Babu, B. V., Angira, R. Optimization of Non-Linear Chemical Processes Using Evolutionary Algorithm, Proceedings of International Symposium & 55th Annual Session of IIChE (CHEMCON-2002), OU, Hyderabad, December 19-22, 2002.

72. Storn, R., Price, K., Minimizing the Real Functions of the ICEC'96 Contest by Differential Evolution in IEEE Conference on Evolutionary Computation, Nagoya, 842-844, 1996.

73. Boroujeng, B. F., Adaptive Filters: Theory and Applications, John Wiley&Sons, 1999.

74. Lopez Cruz I.L., Van Willigenburg L.G., Van Straten, G., Parameter Control Strategy in Differential Evolution Algorithm for Optimal Control, In: M.H. Hamza, Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing, Cancun, Mexico, 211-216, May 21-24, 2001.

75. Yiğit, N., Karaboğa, N., Farksal Gelişim Algoritması İle Adaptif Süzgeçlerde Gürültü Giderme, Sinyal İşleme ve İletişim Uygulamaları Kurultayı SİU 2007, 2007.

76. Yiğit, N., Karaboğa, N., Gürer, B., Adaptif Süzgeçlerde Farksal Gelişim Algoritması Kullanılarak Gürültü Giderme, İTÜSEM07, 2007 (Gönderildi).

77. Yiğit, N., Koyuncu, C.A., Karaboğa, N., Farksal Gelişim Algoritmasının Kontrol Parametrelerinin Adaptif Lineer Toplayıcı Tasarımına Etkisinin İncelenmesi, Sinyal İşleme ve İletişim Uygulamaları Kurultayı SİU 2007, 2007.

EK–1

```
clc;
clear all;
close all;

refGain = 1;
worder =2;

N = 2048;
t=1:N;

signal1= sin(2*pi.*t.*t/N/N*8);
%signal2 = sin(2*pi.*t.*t/N/N*8)+sin(2*pi.*t.*t/N/N*4);
%signal3=sin(2*pi.*t.*t/N/N*8)+sin(2*pi.*t.*t/N/N*4)+sin(2*pi.*t.*t/N/
N*2);

%Gürültünün eklenmesi;

wreal = randn(1,worder);
additivenoise = randn(1,N);

ref = conv(additivenoise,wreal);
primary =signal + ref(1:length(signal));
fref = additivenoise*refGain; %real reference mic

% FG Algoritması

birey=100;
firderece=2;
katsayi=0.4;
f=0.5;
CR=0.8;
maxgener=N;

frefpad = [zeros(1,worder -1) fref];

w(:,:) = randn(birey,worder);
w(1,:) = ones(1,worder);
```

```
w1=w;
w2=w;
start = flops;

for n = 1:maxgener;
     eniyi=1;

    for k=1:birey
        cozum=w(k,:);
        r1=round((rand*(birey-1))+1);
        r2=round((rand*(birey-1))+1);
        r3=round((rand*(birey-1))+1);
            while (k==r1|k==r2|k==r3|r1==r2|r2==r3|r1==r3)
            r1=round((rand*(birey-1))+1);
            r2=round((rand*(birey-1))+1);
            r3=round((rand*(birey-1))+1);
            end

        yenicozum=w(k,:)+katsayi*(w(r3,:)-w(k,:))+f*(w(r1,:)-w(r2,:));

% Çaprazlama ve Yeniden Birleşme

        rs=rand;
        if rs<CR
            adaycozum=yenicozum;
        else
            adaycozum=w(k,:);
        end
        m = n + worder -1;
        frefblock = frefpad(m-worder+1:1:m)';
        cikiseski(n) = w(k,:)*(frefblock);
        cikisyeni(n)=adaycozum*(frefblock);

%Seçme

        if abs(cikisyeni(n)-primary(n))<abs(cikiseski(n)-primary(n))
            w(k,:)=adaycozum;
        end
        if  sqrt(sum(abs((cikisyeni(n)-
primary(n)).^2)))<sqrt(sum(abs((cikiseski(n)-primary(n)).^2)))
```

```
            eniyi=k;
         end
       cikis1(n)=w(eniyi,:)*(frefblock);
       output(n)=(sum((primary(n)-cikis1(n))));
       end

        cikis=cikis1(:)
        weniyide(:,:)=w(eniyi,:)
end;

%%%LMS

mu = .1;
frefpad1 = [zeros(1,worder -1) fref];

for n1 = 1:N;
      m1 = n1 + worder -1;
   frefblock1 = frefpad1(m1-worder+1:1:m1)';
   refP1(n1) = w1(n1,:)*(frefblock1);
   output1(n1) = primary(n1) - refP1(n1);
   w1(n1+1,:) = w1(n1,:) + mu.*frefblock1'.*output1(n1);
end;

work = flops-start
w1(length(w1),:)
weniyilms(:,:)=w1(eniyi,:)

%%%NLMS
mu = .1;
frefpad2 = [zeros(1,worder -1) fref];

for n2 = 1:N;
   m2 = n2 + worder -1;
   frefblock2 = frefpad2(m2-worder+1:1:m2)';
   refP2(n2) = w2(n2,:)*(frefblock2);
   output2(n2) = primary(n2) - refP2(n2);

w2(n2+1,:)=w2(n2,:)+mu/(0.001+frefblock2'*frefblock2)*frefblock2'*outp
ut2(n2);
  end;
```

```
w2(length(w2),:)
weniyinlms(:,:)=w2(eniyi,:)

figure(1);plot(frefpad);
title('gürültü işareti');

figure(2);
subplot(3,1,1); title('gürültülü giris');
plot(primary);
subplot(3,1,2); title('sistem cikisi');
plot(output);
subplot(3,1,3);
plot(signal); title('bilgi isareti');

figure(3);
subplot(3,1,2);
plot(output1);axis([0 length(primary) min(signal)-.5 max(signal)+.5]);
title('LMS filtered output');

figure(4);
subplot(3,1,2);
plot(output2);axis([0 length(primary) min(signal)-.5 max(signal)+.5]);
title('NLMS filtered output');

sv = 2*worder;
sw = length(signal);
SNRpre = norm(signal(sv:sw))/norm(ref(sv:sw));
SNRpre = 10*log10(SNRpre)
```

ÖZGEÇMİŞ

Nalân YİĞİT, 1981 yılında Kayseri'de doğdu. İlk ve orta öğrenimini 60. Yıl Cumhuriyet İlköğretim Okulu'nda, lise öğrenimini Sümer Lisesi'nde tamamladıktan sonra 1998 yılında Erciyes Üniversitesi Elektronik Mühendisliği Bölümünü kazandı. 2003 yılında bu bölümden mezun oldu. 2005 yılında Erciyes Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'ında yüksek lisans öğrenimine başladı. Halen aynı anabilim dalında öğrenimini sürdürmektedir.

Adres: Erciyesevler Mahallesi
Mazı Sokak
Safa Apartmanı
No:11/11
38020 Kocasinan
KAYSERİ

Tel: +90 352 233 24 35
Email: nalanygt@yahoo.com

78.