T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ
YÜKSEK LİSANS PROGRAMI

PCM HABERLEŞME SİSTEMLERİNDE SIKIŞTIRMA ALGORİTMALARININ BAŞARIM ANALİZİ

YÜKSEK LİSANS TEZİ

Hazırlayan
Hakan KAHRAMAN

Danışman
Yrd. Doç. Dr. Soner ÖZGÜNEL

İstanbul – 2015

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ
YÜKSEK LİSANS PROGRAMI

PCM HABERLEŞME SİSTEMLERİNDE SIKIŞTIRMA ALGORİTMALARININ BAŞARIM ANALİZİ

YÜKSEK LİSANS TEZİ

Hazırlayan
Hakan KAHRAMAN

Danışman
Yrd. Doç. Dr. Soner ÖZGÜNEL

İstanbul – 2015

TEZ ONAY SAYFASI

ÖNSÖZ

Tez araştırmalarım süresince her türlü yardım ve fedakârlığı sağlayan, bilgi, deneyim ve güler yüzü ile çalışmama önderlik eden, ayrıca bu çalışmayı bana uygun görerek kendimi geliştirmeye yönelik birkaç adım ileride olmamı sağlayan, tez danışmanım Sayın Hocam Yrd. Doç. Dr. Soner Özgünel'e teşekkürlerimi sunarım.

Tezimin hazırlanması sırasında beni cesaretlendiren ve manevi destek sağlayan değerli arkadaşlarım Can Atılgan, Mehmet Akkavak, Simge Fidan, Cansu El, Büşra Atasever, Sena Derman ve Nuray Işık'a teşekkürü bir borç bilirim.

Bu çalışmayı, derlememde emeği geçen ve benden maddi, manevi hiçbir desteği esirgemeyen aileme şükranlarımı sunarım.

İstanbul, 2015 Hakan KAHRAMAN

İÇİNDEKİLER

KISALTMALAR

FM	: Frekans Modülasyonu	Frequency Modulation
GM	: Genlik Modülasyonu	Amplitude Modulation
DM	: Delta Modülasyonu	Delta Modulation
AWGN	: Beyaz Gauss Gürültüsü	Additive White Gaussian Noise
PCM	: Darbe Kod Modülasyonu	Pulse Code Modulation
PAM	: Darbe Genlik Modülasyonu	Pulse Amplitude Modulation
PWM	: Darbe Genişlik Modülasyonu	Pulse Width Modulation
PPM	: Darbe Konum Modülasyonu	Pulse Position Modulation
PDM	: Darbe Yoğunluk Modülasyonu	Pulse Density Modulation
AM	: Genlik Modülasyonu	Amplitude Modulation
PSK	: Faz Kaydırmalı Anahtarlama	Phase Shift Keying
FSK	: Frekans Kaydırmalı Anahtarlama	Frequency Shift Keying
ASK	: Genlik Kaydırmalı Anahtarlama	Amplitude Shift Keying
QPSK	: Dörtlü Faz Kaydırmalı Anahtarlama Quadrature Phase Shift Keying	
WAV	: Dalga Formatlı Ses Dosyası	Waveform Audio File Format
MP3	: MPEG-1 Ses Katmanı II	MPEG-1 Audio Layer II
PNG	: Taşınabilir Ağ Grafikleri	Portable Network Graphics
JPEG	: Birleşik Fotoğraf Uzmanları Grubu Joint Photographic Experts Group	
ADC	: Analog Dijital Dönüştürücü	Analog Digital Converter

SN : İşaret Gürültü Oranı Signal to Noise Ratio

CCITT1 : Uluslararası Telefonlar Komitesi A.Ş.

Comite Concultaif Internationale de Telegraphie ve Telephonie

ŞEKİLLER

Sayfa No.

GENEL BİLGİLER

Adı ve Soyadı : Hakan KAHRAMAN

Anabilim Dalı : Elektronik ve Haberleşme Mühendisliği

Programı : Elektronik ve Haberleşme

Tez Danışmanı : Yrd. Doç. Dr. Soner ÖZGÜNEL

Tez Türü ve Tarihi : Yüksek Lisans – Ocak 2015

ÖZET

Bu çalışma, günümüzde kullanılan PCM Haberleşme Sistemlerinin sıkıştırma algoritmalarının performanslarını irdelemek amacı ile yapılmıştır.

İlk bölüm, Sayısal ve Analog Haberleşme Sistemlerinde kullanılan yöntemleri içermektedir. Genel olarak sistemler ile ilgili bilgiler verilmiştir. Ayrıca, Sayısal Haberleşme sistemleri ve sıkıştırma yöntemleri incelenmiştir.

Analizlerin yapıldığı bölümde, PCM uygulamalarında kullanılan Örnekleme, Sıkıştırma, Kuantalama, Kodlama ve Hata Grafiklerinin incelenmesi konuları işlenmiştir. Sıkıştırma algoritmaları olarak A-Kuralı, μ- Kuralı ve Düzgün-Olmayan sıkıştırma yöntemleri kullanılmıştır. Yapılan analizlerin tümü Matlab Programlama dili kullanılarak sanal ortamda elde edilmiştir.

Araştırma sonucunda günümüzde kullanılan Sayısal Haberleşme Sistemlerinden biri olan PCM yönteminin farklı fonksiyonlar ve uygulamalar altında nasıl davrandığı belirlenmiştir. Sıkıştırma algoritması kullanılarak kuantalama yapılan sistemlerin daha az hata oranına sahip olduğu belirlenmiştir. Hata grafikleri bir bütün olarak irdelendiğinde, örnekleme sıklığı, bit sayısı ve sıkıştırma yönteminin çok önemli olduğu saptanmıştır.

Anahtar Kelimeler: Kuantalama, PCM, Örnekleme, Sayısal Haberleşme, A-Kuralı

GENERAL INFORMATION

Name and Surname	: Hakan KAHRAMAN
Field	: Electronics and Communication Engineering
Program	: Electronics and Communication
Supervisor	: Assist. Prof. Dr. Soner ÖZGÜNEL
Degree Awarded and Date	: Master of Science – January 2015

ABSTRACT

The main aim of this research is to investigate the compression algorithms of PCM communication systems and analysis methods used nowadays.

First part includes the methods of digital and analog communication systems. In general, they are given information about the system. In addition, Digital Communication Systems and compression methods are examined.

During the analysing process part, Sampling Applications, Compression, Quantization, Coding and Analysis of error graphics issues that are used in PCM techniques are handled. A-law, μ-law and non-linear compression methods are used for compression. All of the analyses are obtained in a virtual platform using MATLAB programming language.

In consequence of the research, how the PCM method, one of the digital communication systems, behaves under the different functions and applications is determined. It is determined that the systems made quantization using the compression algorithm have less error rate. When the error graphics are examined as a whole, the sampling frequency, the number of bits and compression method are found to be very important.

Keywords: Quantization, PCM, Sampling, Digital Communications, A-Law

1. GİRİŞ

Bu tez çalışması, günümüzde kullanılan sayısal ve analog haberleşme sistemlerinin çalışma prensiplerini içermektedir. Sayısal haberleşme sistemleri; darbe genlik modülasyonu (PAM), darbe genişlik modülasyonu (PWM, PDM), darbe konum modülasyonu (PPM), darbe kod modülasyonu (PCM) ve delta modülasyonunu (DM) içermektedir. Analog haberleşme sistemleri; genlik modülasyonu (GM, batı dillerinde AM), faz modülasyonu (PM) ve frekans modülasyonunu (FM) içermektedir. Sayısal haberleşme sistemleri tüm başlıkları ile incelenirken analog haberleşme sistemleri ile ilgili kısa bilgi verilmiştir.

Yöntemlerin incelenmesi sırasında, PCM uygulamalarında kullanılan örnekleme, sıkıştırma, kuantalama, kodlama ve hata grafiklerinin incelenmesi konuları işlenmiştir. Sıkıştırma algoritmaları olarak A-Kuralı, μ- Kuralı ve düzgün-olmayan sıkıştırma yöntemleri kullanılmıştır. Ayrıca sıkıştırma algoritmaların matematiksel olarak kullanımları da ele alınmıştır.

Analizlerin yapıldığı simülasyon bölümünde ise Matlab ortamında PCM uygulaması yazılım aracılığı ile oluşturulmuştur. Matlab yazılımının tercih edilme sebebi ise grafiklerin çok daha basit ve başarılı olarak elde edilmesidir. Matlab analizlerinde, kuantalama yöntemleri, sıkıştırma algoritmaları, hata grafikleri ve kuantalama hataları ele alınmıştır.

Bu çalışmada, yapılan analizler sonucunda sıkıştırma algoritması kullanılarak yapılan kuantalamaların daha az hata yaptığı, bit sayıları arttırıldıkça yapılan hataların azaldığı, doğrusal kuantalama yöntemleri yerine doğrusal olmayan kuantalama yöntemleri kullanıldığında daha başarılı sonuçlar elde edildiği görülmektedir.

2. HABERLEŞME SİSTEMLERİ

Haberleşme, iletilen bilginin (işaret, ses, görüntü) gönderen ve alıcı tarafından ortak bir dilde anlaşıldığı ortamda, bilginin bir göndericiden bir alıcıya iletilme sürecidir. Haberleşmedeki temel prensip, gönderici ve alıcı tarafından kullanılan dilin her iki tarafta da bilinen kaynaklara dayanmasıdır. Belirli işaretlerin sayısallaştırılarak bir haberleşme kanalı aracılığıyla bir kaynaktan bir hedefe/alıcıya iletilmesi haberleşmenin en temel prensibidir (Vikipedia, 2014).

Haberleşme sistemleri, Sayısal Modülasyon ve Analog Modülasyon sistemleri olarak iki ana başlık altında incelenir. Modülasyon, bir taşıyıcı sinyal ile bilgi sinyalini birleştirerek bir kanal üzerinde aktarılması için uygun forma getirme olayıdır ve haberleşme teknolojisinde kullanılan bir yöntemdir (Karakuş, 2011).

Modülasyon için iki temel yöntem vardır:

1. Analog Modülasyon

2. Sayısal (dijital) Modülasyon

Sayısal Modülasyon yöntemleri, başta bilişim ve haberleşme teknolojisi olmak üzere çok çeşitli bir uygulama alanına sahiptir (PSK, FSK, ASK, QPSK gibi yöntemler). Ancak, karasal yayıncılıkta Analog Modülasyon yöntemleri (AM, FM gibi yöntemler) de kullanılmaktadır.

Analog yöntemler şu şekilde sınıflandırılabilir:

a. Genlik Modülasyonu (GM, Batı dillerinde AM)

b. Faz Modülasyonu (PM)

c. Frekans Modülasyonu (FM)

Sayısal yöntemler şu şekilde sınıflandırılabilir:

a. Darbe genlik modülasyonu (PAM)

b. Darbe genişlik modülasyonu (PWM, PDM)

c. Darbe konum modülasyonu (PPM)

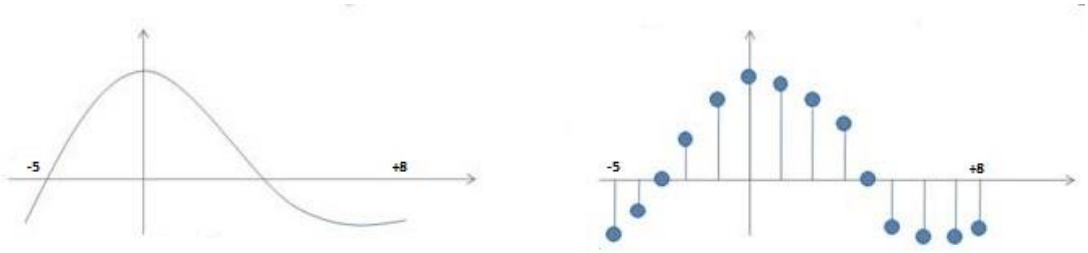
d. Darbe kod modülasyonu (PCM)

e. Delta Modülasyonu

Tüm iletişim sistemlerindeki bilgi taşıyıcı sinyaller verici, iletim ortamı, iletim ortamı kayıpları ve alıcı gibi devre elemanlarından geçerek iletimini tamamlar. Sinyallerin aktarıldığı bu yol boyunca en az kayıp ile hareket edebilmeleri için modülasyon sistemlerine ihtiyaç duyulmaktadır. Bir bilgi işaretinin iletilebilmesi için ihtiyaç duyulan anten boyu, sinyalin dalga boyunun katları olmak zorundadır. Kullanılan anten boyları genellikle λ/2, λ/4 ve λ/8 uzunluktadır. İletilmek istenen bilgi işaretinin frekansı düşük olduğu için dalga boyları çok daha büyüktür. Bilgi işaretini bu kadar büyük dalga boyları ile iletebilmek için kullanılacak anten boyları da çok büyük olmak zorundadır. Pratik hayatta çok büyük anten boyutları ile çalışmak neredeyse imkânsızdır. Anten boyları küçültmek amacı ile bilgi sinyalini kendisinden çok daha yüksek frekanslı bir taşıyıcı sinyal ile modüle etmek iletimini sağlamak için gerekli bir yöntemdir. Bu sebepten dolayı iletişim sistemlerindeki Modülasyon uygulamalarının gerekliliği ve önemi ortaya çıkmaktadır (MEB, 2011).

2.1. Sayısal Haberleşme

Haberleşme, her türlü bilgi işaretinin bir yerden bir yere aktarılması veya değiş tokuşu olarak tanımlanmaktadır. Çeşitli ortamlar veya mesafeler üzerinden dijital sistemleri kullanarak bilgi işaretinin aktarımının sağlanmasına ise elektronik anlamda haberleşme denilmektedir. Bir zaman aralığının tümü yerine sadece belirli zaman anlarında tanımlanmış ve sadece belirli sayısal değerleri alabilen işaretler sayısal işaretler olarak adlandırılmaktadır. Sayısal sinyal, sayısal veri, dijital sinyal veya dijital veri, bir sistem tarafından sayısallaştırılmış bir sinyaldir. Bir analog sinyalden belirli zaman aralığında, belirli kesitlerin örnekleri alınarak analog sinyalin gerçekte tam karşılığı olmayan dijital sinyaller oluşturulur. Bir analog sinyalin belirli zaman aralığında dijital örneklerinin alınması Şekil 2.1'de görülmektedir.



Şekil 2.1 Analog ve dijital sinyal

Bir dijital veri ile analog veri arasındaki en büyük fark, analog sinyal sürekli ve değişen ölçekler arasında hareket etmektedir, dijital bir sinyal ise belirli örnekleme zamanlarında ve sayısal ölçekler arasında sınırlı olmasıdır. Oluşturulan dijital veriler, çeşitli rakamların arka arkaya sıralanması ile elde edildiği için üzerinde genlik değişimi, sıkıştırma, genişletme gibi değişiklik işlemleri yapılabilir. Dijital verilerdeki yapısal değiştirme işlemleri çeşitli uygulamalarda görülmektedir:

i. Sayısal ses ortamında, WAV biçimindeki bir ses MP3 biçimine çevrilebilir.
ii. Sayısal resim ortamında, PNG biçimindeki bir resim JPEG biçimine çevrilebilir.
iii. Herhangi bir veri, RSA kullanılarak şifrelenebilir.

Sayısal verinin bir diğer avantajı ise sayısal veriye şifreleme kodları veya hata düzeltme kodları eklenerek verinin şifrelenmesi veya bozulan kısmının belirli bir yüzde içinde tamir edilebilmesi sağlanabilir. Bunun yanında, sayısal veri bir iletim hattında gönderilirken başka bir sayısal veri ile aynı ortama konulup daha sonra elde edilebilir bir şekilde gönderilebilir. Analog veriye karşıya tüm bu avantajlar doğrultusunda, sayısal veri birçok alanda (karasal, dijital, mobil telekomünikasyon, İnternet, film ve müzik saklaması gibi) analog veri uygulamalarının yerini almaktadır (Vikipedia, 2009).

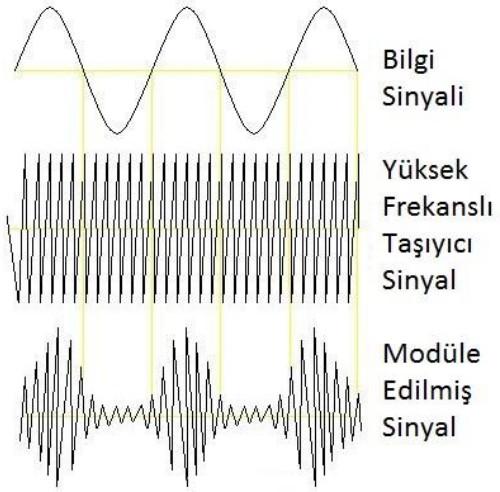
2.2. Modülasyon

Modülasyon, bilgi sinyalinden çok daha yüksek frekanslı bir sinyalin çeşitli özelliklerinin iletilmek istenen bilgi sinyaline bağlı olarak bir araya getirilmesi olayıdır (MEB, 2013). Kodlanma adıyla da bilinmektedir. Modülasyon işlemi

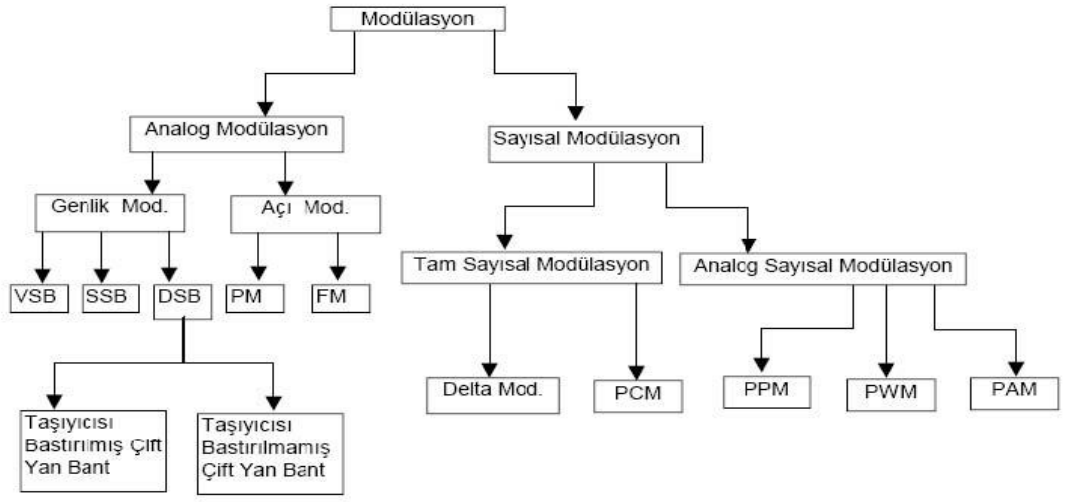
sırasında yüksek frekanslı olarak tanımlanan sinyale, taşıyıcı sinyal denilir. Taşıyıcı sinyal genellikle sinüs veya darbe sinyalidir. İletilen bilginin türüne ve taşıyıcı sinyalin durumuna göre çeşitli modülasyon işlemleri uygulanmaktadır. Şekil 2.2'de Modülasyon örneği görülmektedir.

Modülasyon işleminin başlıca yaraları şunlardır:

i. Yayılımı ve direnci arttırır. Elektromanyetik sinyaller yaklaşık olarak ışık hızında yayıldığı ve uygun koşullar altında dağ, tepe, çukur, suni yapılar gibi engelleri kolaylıkla aşarlar. Uzayda ortamında ise uygun bir anten kullanılarak çok uzaklara gidebilirler.
ii. Sinyal üzerindeki gürültü ve bozulmayı azaltır.
iii. Sinyaller arası kanal ayrımına imkân sağlar. Modülasyon uygulanmış sinyaller ile aynı iletim hattında birden çok bilgi yollanabilir.
iv. Uygulanılan modülasyon işlemi sinyalin frekansını arttırdığı için dalga boyuna bağlı olarak kullanılacak anten boyları küçülür.
v. Modülasyon işlemi sonucu elde edilen sinyal daha kararlı hale geldiği için çevresel etkiler ve sınırlayıcı faktörler azalır.



Şekil 2.2 Modülasyon uygulanmış bir sinyal



Şekil 2.3 Modülasyon türleri

Modülasyon işlemi Analog Modülasyon ve Sayısal Modülasyon olarak uygulanmaktadır. Şekil 2.3'te modülasyon sistemleri görülmektedir.

2.2.1 Sayısal Modülasyon

Sayısal haberleşme, bir bilgi işaretin iletişim kanalları üzerinden sürekli dalga formu yerine belirli zaman konumunda ki sayısal veriler (1 ve 0'lar) olarak iletilmesidir (MEB, 2012). Bilginin kaynağı analog bir sistemden geliyor ise Analog/Dijital (A/D) dönüştürücü ile analog veriler sayısal verilere çevrilir. Bilgi kaynağında üretilen sinyal sayısal veri ise direk olarak kullanılabilir. Bir iletim kanalı üzerinden bir yerden bir yere aktarılacak olan işaretin güçlendirilip analog sinyallere dönüştürülmesi işlemi Sayısal Modülasyondur. Bilişim ve üretim teknolojisindeki gelişmeler sürecinde günümüzde üretilen dijital sistemlerin analog sistemlere göre çok daha kararlı ve ucuz olması sebebiyle Sayısal Modülasyon sistemleri tercih sebebi olmaktadır. Uygulanan sayısal modülasyon işlemlerinin, analog modülasyon işlemlerine göre birçok avantaja sahiptir.

Sayısal modülasyonun avantajları:

i. Sayısal modülasyon işleminde iletilen güç, yalnız kısa işaretler içinde yoğunlaşmıştır. Analog modülasyon işlemindeki gibi sürekli olarak farklı genliklerde dağılmamıştır. Elde edilen bu güç dağılımının darbeler üzerinde olması sistem tasarımcılarına önemli kolaylıklar sağlamaktadır. Yüksek güçlü mikrodalga tüpleri, lazerler sistemler darbe biçiminde çalışmaya elverişli eleman örnekleridir.
ii. İletilen bilgiler arasındaki darbe boşlukları belirli kodlama yöntemleri ile diğer bir bilgiye ait işaretler ile doldurularak aynı iletim kanalı üzerinden birden fazla bilgi taşımak mümkündür.
iii. Sayısal Modülasyon işleminin avantajlarının son yıllarda ön plana çıkmasından dolayı tercih edilen bilgilerin darbe sinyalleri olarak üretiminin artması, sayısal haberleşme sistemlerinin de üretiminin gelişmesini sağlamıştır.
iv. Analog sinyallere göre daha avantajlı olan Sayısal işaretlerin yaygınlığının artması, sistemlerde kullanılmakta olan teknikleri de yaygınlaştırmaktadır.
v. Sayısal sistemlerde kullanılan darbeler, gürültü ve bozulmalara daha dayanıklı olması sebebi ile bazı haberleşme sistemlerinde tercih edilmektedir.

Sayısal modülasyonun dezavantajları:

i. Sayısal bir haberleşme sistemde kullanılan bant genişliği, aynı bilginin iletilmesi için kullanılan analog bir sisteme ait bant genişliğinden çok daha fazla olmak zorundadır.
ii. İletilmek istenen analog sinyaller, vericiye ulaşmadan önce dijital kodlara ve alıcı tarafında ise tekrar analog sinyale dönüştürülmelidir.
iii. Kullanılan saat darbelerinin frekansları ve zamanlamaları tüm sayısal sistemler tarafında aynı yani senkronize olmalıdır.
iv. Günümüzde kullanılmakta olan analog haberleşme sistemi donanımları ile sayısal haberleşme sistemi donanımları uyumlu değildir.

2.2.2 Sayısal Modülasyon Yöntemleri

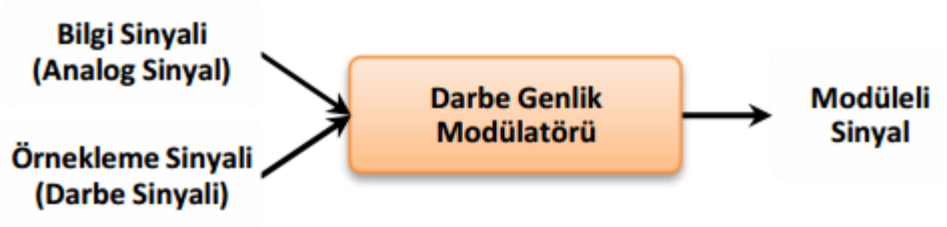
Sayısal haberleşme sistemlerinde beş çeşit modülasyon tekniği kullanılmaktadır. Modülasyon teknikleri:

1. Darbe genlik modülasyonu (PAM)
2. Darbe genişlik modülasyonu (PWM, PDM)
3. Darbe konum modülasyonu (PPM)
4. Delta Modülasyonu
5. Darbe kod modülasyonu (PCM)

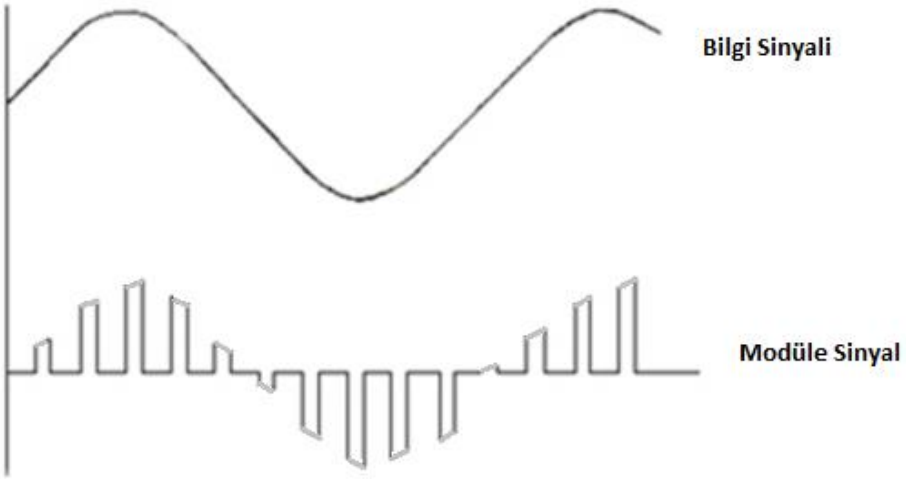
2.2.2.1 Darbe Genlik Modülasyonu (PAM)

İletilmek istenen bilginin, kare dalgalardan oluşan taşıyıcı sinyali ile bilgi sinyalinin genliğine (amplitude) bağlı olarak düzenlenmesine "Darbe Genlik Modülasyonu" (Pulse Amplitude Modulation - PAM) denir. Şekil 2.4'te PAM şeması gösterilmektedir.

Taşıyıcı kare sinyal ile bilgi sinyalinin çarpılması Darbe Genlik Modülasyonun temel mantığıdır. İki sinyalin çarpılarak elde edilmesi yöntemine tabi örnekleme yöntemi denir. Darbe Genlik Modülasyonunda bu şekilde elde edilen sinyalin tepeleri bilgi sinyaline ait formları yansıtır. Şekil 2.5'te tabii örnekleme yöntemi gösterilmektedir.

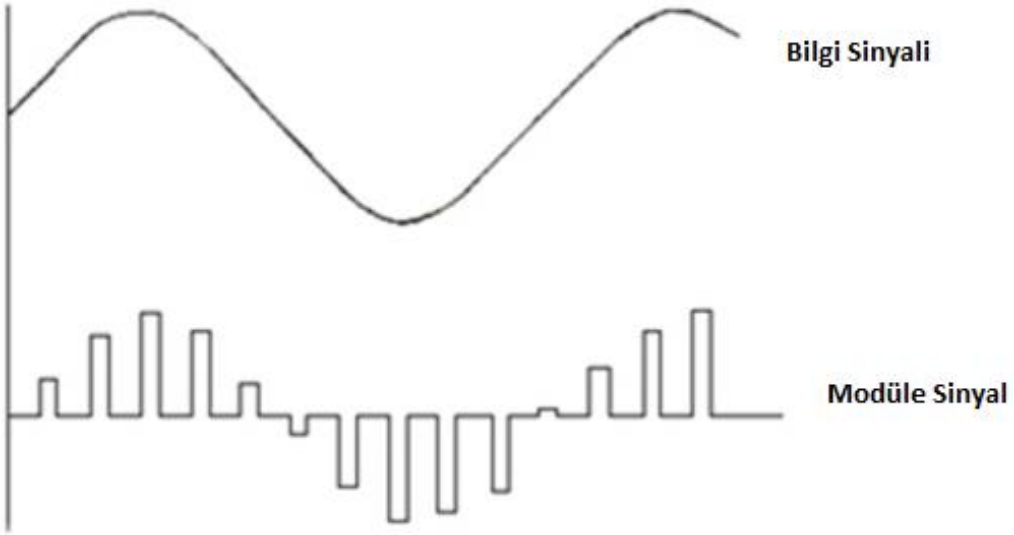


Şekil 2.4 Darbe genlik modülasyonu şeması



Şekil 2.5 Tabii darbe genlik modülasyonu uygulanmış sinyal

Darbe Genlik Modülasyonunda, tabii örneklenmiş modülasyon yöntemi haricinde düz tepe (flat-top) örneklenmiş modülasyon yöntemi de kullanılmaktadır. Düz tepe örnekleme yönteminde, darbeler (pulse) örnekleme süresince bilgi sinyalinin genliğini kesin hatlar ile takip etmez. Örnekleme işlemi süresince, anlık tek bir genlik değerine sahiptir. Genlik değerinin elde edilmesi, örnekleme aralığında (genellikle başlangıçta) özel bir konumda bilgi sinyali ile eşdeğer değere sahip dikdörtgen darbelerden oluşur. Şekil 2.6'da düz tepe örnekleme yöntemi örneği gösterilmektedir.



Şekil 2.6 Düz tepe darbe genlik modülasyonu uygulanmış sinyal

İki örnekleme yöntemi kullanılarak alınan örnekler, düz tepe örneklenmiş darbe genlik modülasyon sinyali ile elde edilmiş bilgi sinyali, tabii örneklenmiş darbe genlik modülasyon sinyali ile elde edilmiş bilgi sinyaline göre çok daha fazla distorsiyonludur. Elde edilen örneklenmiş sinyallerdeki darbe genişliği (W), örnekleme periyodu olan (T)'ye göre azaldıkça distorsiyon değeri de azalır. Sinyalin darbe genişliğinin büyük olma problemi, karşı tarafta yeniden elde edilen bilgi sinyalinin bir filtre devresinden geçirilmek suretiyle distorsiyon problemi giderilebilir (MEB, 2013).

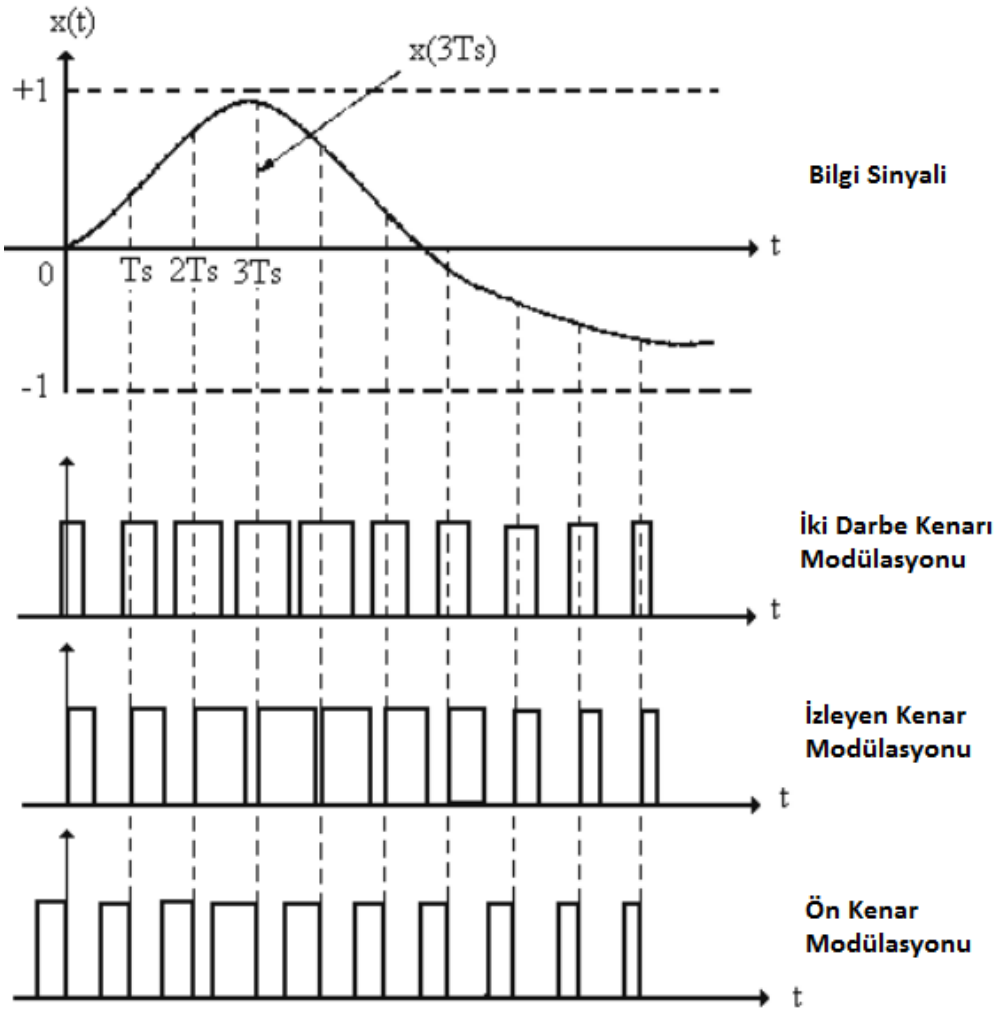
2.2.2.2. Darbe Genişlik Modülasyonu (PWM)

İletilmek istenen bilginin, kare dalgadan oluşan taşıyıcı sinyalinin darbe genişliği (W) olan bilgi sinyalinin genliğine bağlı olarak değiştirilmesine Darbe Genişlik Modülasyonu (Pulse Width Modulation - PWM) veya Darbe Süresi Modülasyonu (Pulse Duration Modulation -PDM) denir. Darbe genlik modülasyonunda elde edilen sinyalin zayıflama ve parazit gibi bozulmalarını engellemek amacı ile darbe genişlik modülasyonu geliştirilmiştir (MEB, 2013).

Üç farklı biçimde Darbe Genişlik Modülasyonu uygulanmaktadır. Örnekleme, sinyal süresinin genişliği ön darbe, arka darbe veya her iki kenar darbesi birden olmak üzere üç farklı biçimde değiştirilebilir. Şekil 2.7'de üç farklı yöntem ile uygulanmış darbe genişlik modülasyonu gösterilmektedir.

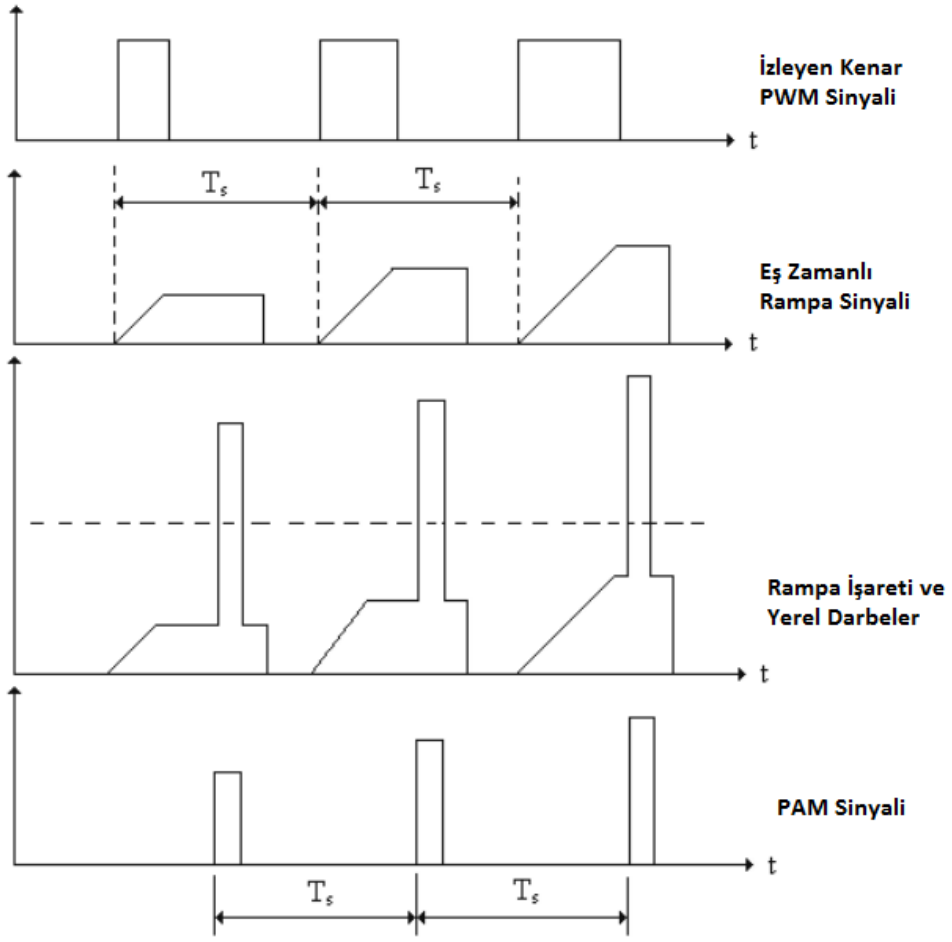
Bilgi sinyali, PWM sinyalinden iki yöntemle elde edilir:

i. Alçak geçiren filtre kullanılarak bant genişliği W olan PWM dalgası filtreden geçirilir. Filtreden geçirilen PWM dalgasının ardışık harmonileri elde edilir. Demodülasyon işleminden sonra oluşan x(t) mesaj işaretlerinin distorsiyonlu olarak elde edilmesi, bu yöntemin önemli bir sakıncasıdır. Spektrumdaki harmonilerin yan bantlarının kuyrukları temel banda kadar uzanmasından kaynaklanan neden bu uygulamanın sebebidir.
ii. PWM sinyali önce PAM sinyal biçimine dönüştürülür ve elde edilen PAM sinyali alçak geçiren bir filtreden geçerek x(t) mesaj işaretinin oluşturulması ise ikinci bir yöntem olarak kullanılmaktadır.



Şekil 2.7 Darbe genişlik modülasyonu uygulanmış sinyal

Bilgi sinyalini elde ederken kullanılan ikinci yöntemde, PWM sinyallerinin ön kenarı kullanılarak bir düzgün rampa işareti üretilir. Sinyaldeki rampaların yükselişi diğer darbenin düşen kenarında ulaştığında sonlanır. Sinyaldeki darbe süreleri direk olarak rampaların yüksekliği ile orantılıdır. Tepelerin oluşumu, rampalarda elde edilen maksimum değerin belirli bir süre daha aynı genlikte kalmasıyla oluşur. Genliği ve periyodu sabit, zamanlama olarak rampanın sabit değerine oturacak biçimde oluşturulmuş bir sinyal alıcı tarafında yapılan demotülatör kısmında oluşturulur. Elde edilen sinyal, rampadaki sinyal ile ara bir işlemde toplanması sonucu birleşir. Elde edilen sinyal üzerindeki istenmeyen yani belirli bir eşik değeri altında kalan parçaları, bir kırpma işlemi sayesinde yok edilir. Tüm işlemlerin sonucunda istenilen PAM sinyaline ulaşılmaktadır. Şekil 2.8'de PAM sinyalinin elde edilmesi gösterilmektedir.



Şekil 2.8 PWM sinyalinin PAM sinyaline dönüştürülmesi

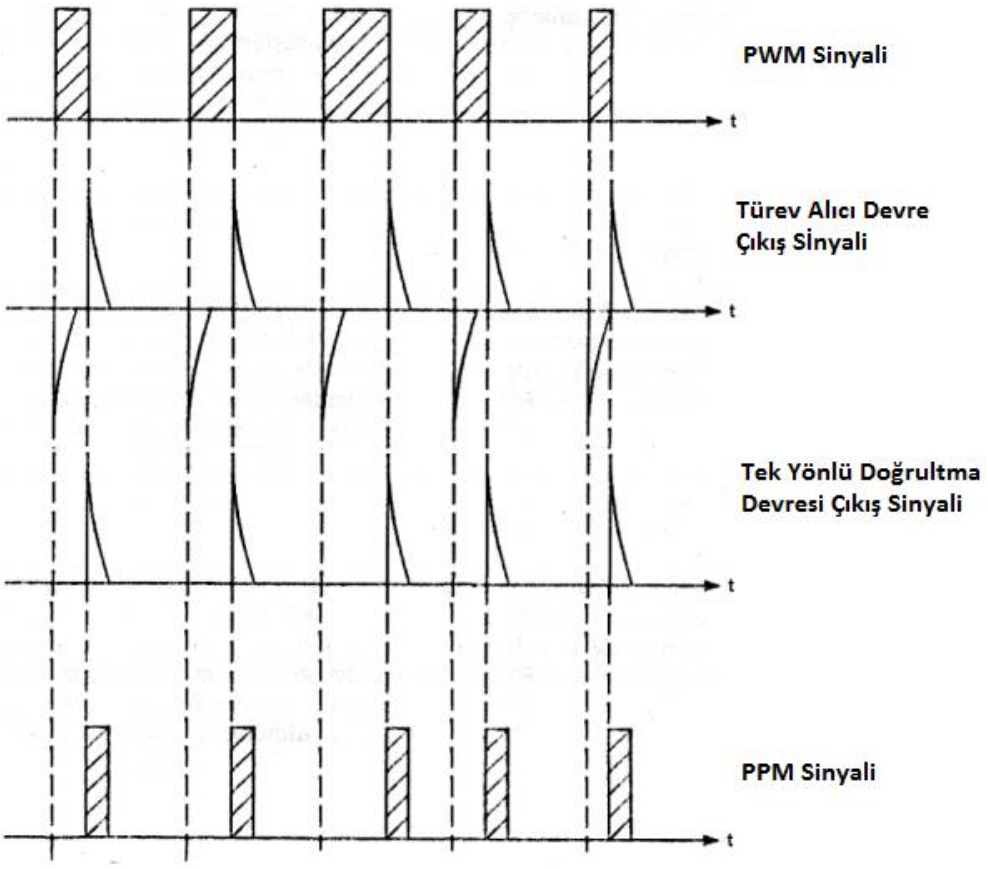
2.2.2.3. Darbe Konum Modülasyonu (PPM)

İletilmek istenen bilginin, kare dalgadan oluşan taşıyıcı sinyalinin, başlangıç konumunu bilgi sinyalinin genliğine bağlı olarak oluşturulmasına Darbe Konum Modülasyonu (Pulse Position Modulation - PPM) denir (Ertürk, 2005).

Uygulanan yöntemlere bakıldığında, darbe konum ve darbe genişlik modülasyonu birbirine çok yakın yöntemler olarak görülmektedir. Darbe genişlik modülasyon sinyalinde, alıcı kısmında çok fazla gürültü oluşması sonucu oluşturulan sinyallerden deformasyonlar görülmektedir. Oluşan deformasyonların engellenmesi amacı ile darbe konum modülasyonu ortaya çıkmıştır. Bu işlemde, izleyen kenar darbe genişlik modülasyon sinyali kullanılarak Darbe Konum Modülasyon sinyali elde edilir.

Oluşturulan Darbe Genişlik Modülasyon sinyalinin tekrar modülasyon işlemine sokulması gereksiz olarak görülse bile, diğer modülasyon uygulamalarına (PAM ve PWM) göre gürültüye karşı çok daha dayanıklıdır. Ancak iki kez uygulanan modülasyon işlemi, alıcı ve verici taraflarında yapılan modülasyon ve tersi uygulaması maliyeti yükseltmektedir.

İlk adımda bilgi sinyali, PWM sinyaline dönüştürülür. Oluşturulan PWM sinyali, türev alıcı bir devreden geçirilir. Türev işlemini yapan devre, PWM sinyal saat darbelerini başlangıç noktaları için eksi (-) saat darbesi ve arka kenarları için artı (+) saat darbesi oluşturur. Oluşturulan PWM sinyalinin genişlikleri, tüm çift saat darbe işareti arasındaki sürelere eşittir. Yarım dalga çevirici (Half Wave Rectifier) devresi kullanılarak, türev alıcı devrede oluşturulan sinyalin eksi bölgede bulunan kısımları kırpılarak sinyalden ayrılır. Yarım dalga çevirme (Half Wave Rectifier) devresi, çıkış sinyalini, sinyal biçimlendirici kullanarak darbe katranına dönüştürülerek Darbe Konum Modülasyon sinyali oluşturulur. Şekil 2.9'da PPM sinyalinin oluşturulması gösterilmektedir.



Şekil 2.9 PWM sinyalinden PPM sinyalinin elde edilmesi

Örnek olarak, sistem t_s periyoduna sahip olduğunda, darbe konum modülasyonunda dikdörtgen işaretlerin konumu, girişte kullanılan bilgi sinyalinin değerlerini göstermektedir. Darbe genlik modülasyonunda gönderilen genlik değerinden ve darbe genişlik modülasyonunda gönderilen sinyal sürelerinden dolayı harcanan enerji, tek başına konum değerinin iletilmesi sebebiyle tüketilemez. Darbe Konum Modülasyonu, Darbe Genlik Modülasyonu ve Darbe Genişlik Modülasyonuna göre daha küçük enerji seviyelerinde çalışmaktadır. Alıcı tarafındaki de-modülasyon işlemi modülasyon işleminin tam tersidir. Sonuç olarak, PWM sinyali PPM sinyalinden önce elde edilir. Sonraki işlemde ise PWM sinyalinden PAM sinyaline geçiş yapılmaktadır.

2.2.2.4. Delta Modülasyonu (DM)

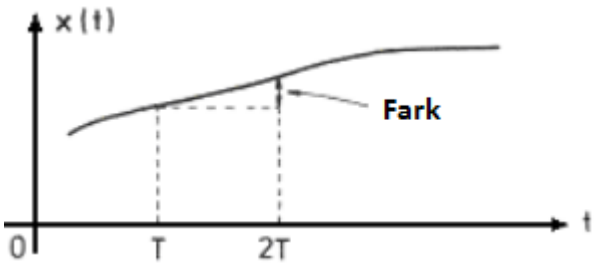
Eğim modülasyonu olarak ta bilinen Delta modülasyonu, PCM gibi diğer bir sayısal sistem modülasyon yöntemidir. x(t) analog bilgi işaretinden belirli kesitlerle elde edilen örnekler kuanta seviyelerine yuvarlatıldıktan sonra dijital kodlanması PCM sisteminin temel prensibidir (Ertürk, 2005).

Görüntü işaretlerinde görüldüğü gibi pratik uygulama işlemlerinde karşılaşılan bazı işaret türlerinde, komşu örnekler arasında önemli bağlantılar vardır. Örnekler arasındaki ilişkilerin büyük olması, bir örneğin kuanta seviyesinin bilinmesi, onu takip eden komşu örneklerin de kuanta seviyelerinin büyük olasılıkla tahmin edilmesini kolaylaştırır. Bu bilgi doğrultusunda, her örneğin kuanta seviyesinin tek tek iletilmesine gerek kalmayabilir. İşaretin gerçek kuanta seviyeleri yerine işaretin komşuları arasındaki değişimler (farklar) belirlenip iletilebilir. Böylece, her işaret için kodlanıp gönderilecek bilgi boyutunda, büyük miktarda azalma ve güç tasarrufu sağlanabilir. DM uygulama işlemi Şekil 2.10'da gösterilmektedir.

DM uygulamasında, komşu işaretler arasındaki genlik (amplitude) değişim farklarının kodlanması gerektiğinden, karşımıza çıkan kodlama yöntemine diferansiyel PCM ve da DPCM denir. Komşu işaretler arasındaki farkın sadece pozitif (+) veya negatif (-) olduğunun saptandıktan sonra yapılan kodlama işlemi DPCM'in en basit yöntemidir. Bu yöntem Delta Modülasyonu olarak bilinmektedir.

Delta Modülasyonu diğer bir ifade yöntemiyle, iki işaret örneği arasındaki genlik farkının birer bit ile ifade edilmesidir. Bu yöntem ile yapılan Delta Modülasyonuna "Doğrusal Delta Modülasyonu" adı verilir.

Genlik değerlerinin çok hızlı değiştiği işaretlerde, doğrusal modülasyon yöntemi uygun değildir. Delta Modülasyonunda değişen genliklerin eğimlerinin yakalanması sınırlı olması yöntemin dezavantajı olarak bilinmektedir. Ancak Delta Modülasyonun eğim yakalama kabiliyeti , sıkıştırma - genleştirme (compression - companding) yöntemleri ile hızlandırılabilmektedir (Çıbuk, 2004).



Şekil 2.10 DM örnekler arasındaki fark

3. DARBE KOD MODÜLASYONU (PCM)

Modülasyon yöntemleri arasında, Darbe Genlik Modülasyonu, Darbe Genişlik Modülasyonu ve Darbe Konum Modülasyonu olarak bilinen yöntemlere aynı zamanda analog darbe modülasyon yöntemleri de denir (MEB, 2013). Bilgi sinyalinin, sayısal olarak kodlanmış bir dizi dijital anlık genlikler (pulse) tarafından oluşturulması ile gerçekleştirilen sayısal modülasyona Darbe Kod Modülasyonu (Pulse Code Modulation) denir. Basit olarak bilgi işaretin, ikilik (binary) sayı sisteminde kodlanması ile oluşan bir modülasyon yöntemidir. Bilgi sinyalinde, önceden tanımlanmış zaman aralığında kodlanacak işaretin var ya da yok olması belirlenerek 1 ya da 0 sayı bilgisi gönderilir. PAM, PWM ve PPM yöntemlerinde elde edilen modülasyon yapılmış sinyaller ikili (binary) sayı sisteminde tanımlı bir işaret olarak kabul edilmezler.

Darbe Kod Modülasyonun da işaretlerin gerçek değerleri gönderilmez. Bilgi sinyalinin örneklenmiş değerleri yerine, onun sayısal karşılığı en yakın tam sayıya yuvarlanarak kodlanır ve gönderilir. Sinyalin gerçek değeri yerine onu tanımlayan sayısal bir değer ile ifade edilmesine "Kodlama" denir.

PAM, PWM ve PPM gibi modülasyon yöntemlerinde, modülasyon yapılmış sinyal parametreleri bilgi sinyaline bağlı olarak şekillenebilir ve bilgi sinyaline göre değişik bir değer alabilir. İletilen sinyal, çeşitli sebepler tarafından bozulmuş veya yok olmuş ise alıcının bozulmuş sinyali tam olarak çözebilmesi ve anlamlı bilgi üretmesi mümkün değildir. Gönderilen sinyalin 0 ve 1'lerden oluşması kararlılığını arttırdığı için bozulmalara karşı direnci artmaktadır.

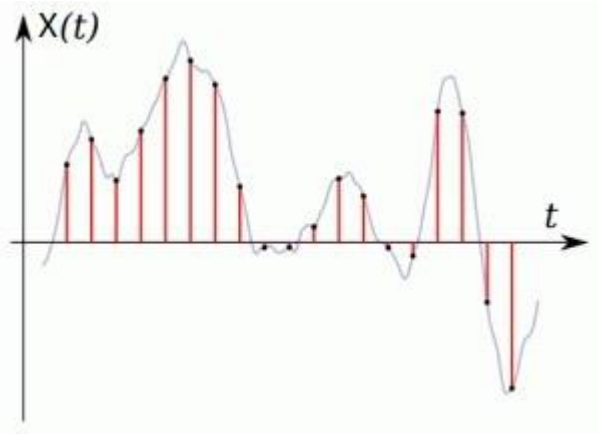
Darbe Kod Modülasyonu; örnekleme - tutma devresi, kuantalama ve kodlama olmak üzere üç temel bölümden oluşur. Şekil 3.1'de blok diyagramı gösterilmektedir.

Şekil 3.1 PCM blok diyagramı

3.1. Örnekleme ve Tutma Devresi

Modülasyon işlemi sırasında, örnekleme ve tutma devresinin amacı, değişen analog giriş sinyalini periyodik olarak belirli kesitler ile örneklemek ve örneklemeyi, art arda sabit genlikli PAM seviyelerine dönüştürmektir. Üretilen sinyaller, sayısal haberleşme sistemlerinde analog bilgi sinyalini, Analog-Sayısal Dönüştürücüler (ADC) yardımıyla sayısal sinyale dönüştürülür. ADC'ye giren bilgi sinyalinin frekansı yüksek olduğu takdirde sinyal üzerindeki değişimlerin çevirici tarafından algılanması ve sayısallaştırması kararlı bir şekilde olmalıdır. Yüksek değişkenliğe sahip analog giriş sinyalinin sabit bir değerde tutulması için kullanılan devrelere Örnekleme ve Tutma Devresi denir. Şekil 3.2'de PCM örnekleme işlemi gösterilmektedir (Karakuş, 2011).

Modülasyon işleminde, Nyquist örnekleme teoremi, bir PCM sistemde kullanılabilecek minimum örnekleme frekans hızını (f_s) olarak gösterir. Örnekleme işleminde, örneklenen sinyalin alıcıda tarafında doğru şekilde tekrar oluşturulabilmesi için, analog giriş sinyalinin (f_a) her döngüsü en az iki kez örneklenmek zorundadır. Bilgi sinyali için minimum örnekleme hızı, en yüksek giriş sinyalinin frekansının iki katına eşittir. Bilgi işaretini örnekleme için kullanılan frekans f_s, f_a'nın iki katından daha küçük ise sinyal üzerinde bozulmalar oluşur. Sinyal üzerinde oluşan bozulmalara katlama bozulmaları denir. İhtiyaç duyulan minimum Nyquist örnekleme hızı, matematiksel olarak, $f_s = 2f_a$ olarak tanımlanır (MEB, 2013).



Şekil 3.2 PCM örnekleme işlemi

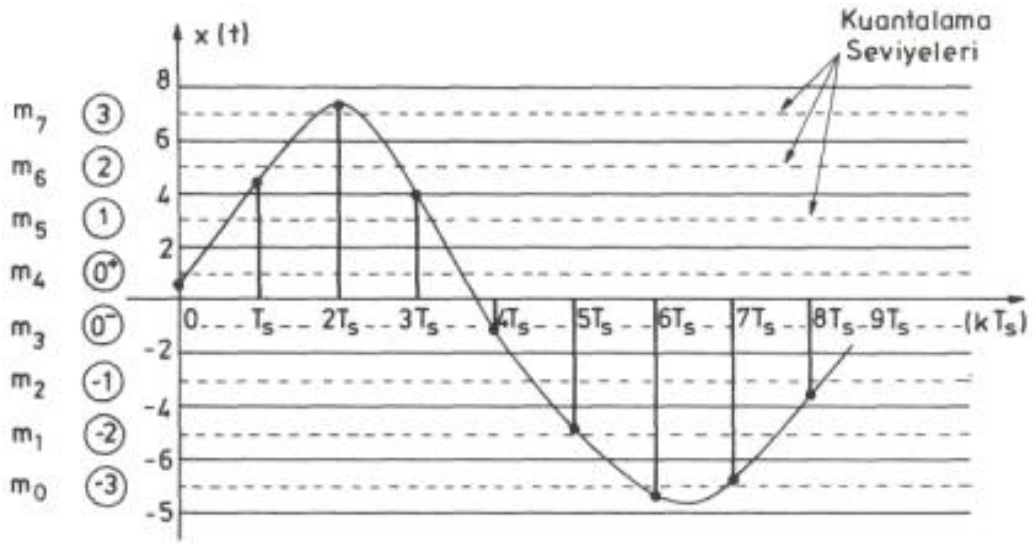
3.2. Kuantalama

Darbe Genlik Modülasyonunda, genlik kesitleri örneklenmiş işaretler belirli kuantalama seviyelerindeki tam sayılara yuvarlandıktan sonra iletilmektedir. Yapılan bu işlem, örneklenen işaretin bozulmalara ve gürültüye olan dayanıklılığı açısından iyileştirme gerçekleştirmeyecektir. Yapılan bu yuvarlama işleminin amacı, bit sayısına bağlı olarak belirlenen kuanta seviyelerinin tam sayı değerlerine karşılık düşecek anlamlı örnek değerleri üretmektir (Ertürk, 2005).

Bilgi işareti olan x(t) sinyalinin maksimum ve minimum genlik seviyeleri +Amax ile -Amax arasında değişiyorsa ve bu aralıkta değişen genlik seviyeleri Q = 2n adet eşit kuanta seviyesine ayrılmak isteniyorsa, a kuantalama aralığı veya adımı:

$$a = \frac{2A_{max}}{2^n}$$

Bilgi sinyali üzerindeki kuantalama işleminde örneklenecek seviyelerin bulunduğu aralık belirlenmektedir. Şekil 3.3'te -8 ve +8 volt arasında değişen bir x(t) bilgi sinyali incelenmektedir. Bu aralık, 8 kuanta seviyesine ayrılırsa, kuantalama aralığı a=(16/8)= 2 birim olacaktır. Şekil 3.3'te örnek kuantalama ve kuanta seviyeleri gösterilmektedir.



Şekil 3.3 Örnek kuantalama seviyeleri

Şekil 3.3'te gösterildiği gibi, her örnekleme değeri 8 kuanta seviyeden birisine yuvarlanmaktadır. Bu örnek için kuanta seviyeleri ±0, ±1, ±2, ±3 olmaktadır. Her örnekleme kesitinde elde edilen değer, en yakın kuanta seviyesinde bulunan tam sayıya kuantalanır. Şekil 3.4'te çeşitli işaret genliklerine karşılık gelen kuanta seviyeleri ve kod dizileri görülmektedir.

Örnekleme işlemi sırasında kuantalama aralığı sayısı Q arttırıldıkça kuantalama hatası da azalmaktadır. Q sayısının artmasını sağlamak için kuantalamada kullanılacak bit sayısını da arttırmak gereklidir.

Örneklenen Sinyal Genliği	Kuanta Aralığı	Kuanta seviyesi
-2.768	-2V , -4V	-1
2.432	2V , 4V	+1
6.536	6V , 8V	+3
-0.025	0V , -2V	-0

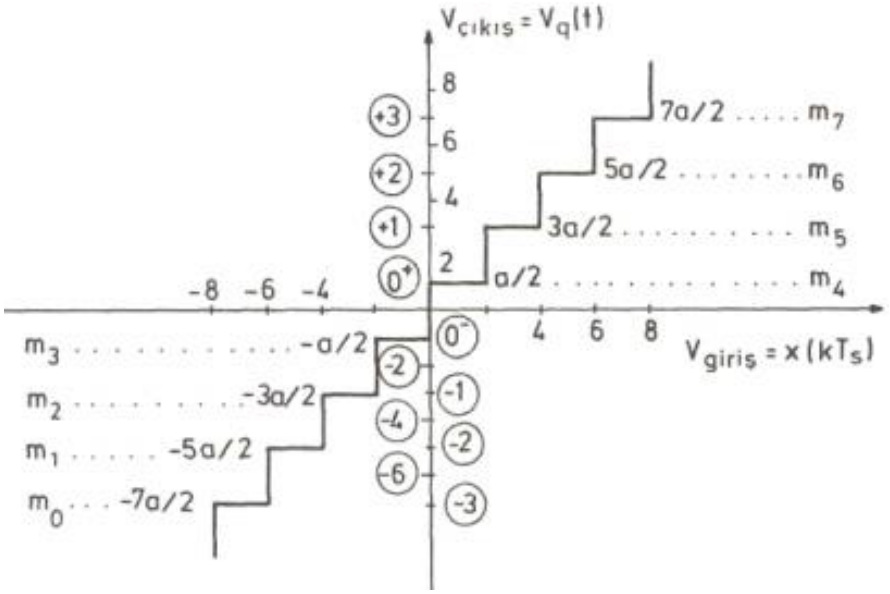
Şekil 3.4 Örnek kuantalama değerleri

Oluşturulan sayısal kodlardan tekrar analog sinyali elde etmek için, PCM sistemin alıcı bölümünde işlemlerin tam tersi yapılarak analog mesaj işareti elde edilir (MEB, 2013).

3.2.1 Düzgün Kuantalama (Lineer)

Kuantalama işlemi sırasında bir sinyalin genliğini eşit aralıklarla parçalayarak eşit seviyeler oluşturulması işlemine düzgün kuantalama denir. Düzgün kuantalama işleminde her bir kuanta aralığı yani Q eşittir (MEB, 2013). Örnek bir sinyal için 8 seviyeli düzgün bir kuantalayıcıya ait giriş-çıkış eğrisi Şekil 3.5'te gösterilmektedir.

-3, -2, -1, -0, +0, +1, +2, +3'teki 8 eşit kuanta seviyesi, sırasıyla m0, m1, m2, m3..., m7 olarak tanımlanmaktadır. Elde edilen seviyeler ikili sistemde bitler olarak kodlanmaktadır. Kodlama işlemi süresince, elde edilen bitlerin en kısa diziler şeklinde olması, sistem performansı için tercih edilir. Verilen örnekte, 8 eşit kuanta seviyesi için $8 = 2^3$ olduğundan, kullanılacak bit kodlama uzunluğu n = 3 olarak hesaplanmaktadır. Şekil 3.6'da kuanta seviyeleri ve ikili kod dizileri gösterilmektedir.



Şekil 3.5 Düzgün kuantalama eğrisi

Kuanta Seviyeleri	Bit Değerleri	Kuanta Seviyeleri	Bit Değerleri
m_0	000	m_4	100
m_1	001	m_5	101
m_2	010	m_6	110
m_3	011	m_7	111

Şekil 3.6 Kuanta seviyeleri ve bit değerleri

Şekil 3.7 PCM düzgün kuantalama blok diyagramı

Modülasyon işleminden sonra darbe dizisi biçiminde oluşan PCM dalgaları, direk olarak bir iletim hattı üzerinden veya analog modülasyon yöntemleri kullanılarak gönderilebilir. Alınan sayısal sinyali, alıcı tarafında tekrar eski analog sinyale dönüştürmek için yapılacak işlem oldukça basittir. Gelen sinyalin şekline veya genliğine bakılmaksızın sadece bir dalganın varlığının veya yokluğunun karar verilmesi yeterli olacaktır. İkili işaret dizisi belirli bir sıra ile oluşturulduktan sonra, kod çözülerek kuantalanmış örnek seviyeler elde edilebilir. Şekil 3.7'de Düzgün PCM blok diyagramı gösterilmektedir.

3.2.2. Düzgün Olmayan (Non-Lineer) Kuantalama

Düzgün kuantalama yönteminde bilgi işareti, en küçük aralıktan daha küçük bir seviyeye sahip ise, çevresel etkenlerden oluşan gürültü işaretten daha büyük olur ve bu Boş Kanal Gürültüsü olarak tanımlanmaktadır (MEB, 2013). Oluşan bu gürültünün yok edilmesi için +0 ve -0 kuanta seviyeleri 0 olarak kabul edilir.

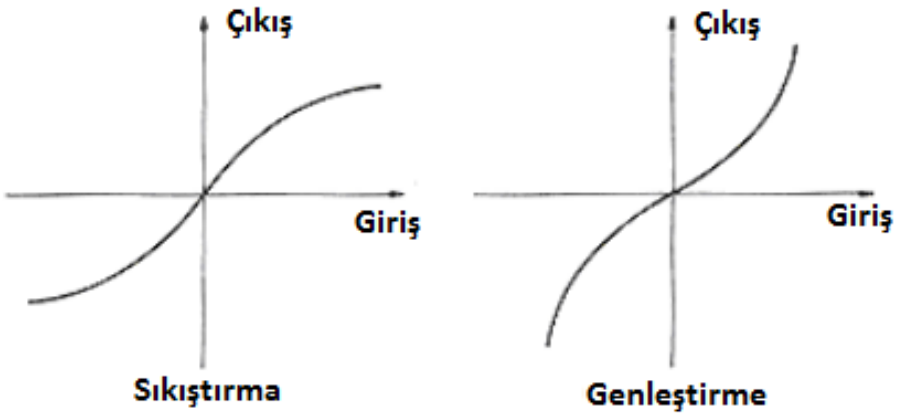
Genel olarak bir ses işaretlerinin dağılımı incelendiğinde, küçük genliklere çok daha fazla rastlandığı görülmektedir. Bu incelemede edinilen bilgi

doğrultusunda, küçük kuanta seviyelerindeki gürültü etkilerinin çok daha fazla olacağı ortaya çıkmaktadır.

Dış etkenlerden gelen gürültüyü azaltmak için izlenecek yöntem, bit seviyesi arttırılarak adım büyüklüğünün azaltılması veya dilim sayısının artırılmasıdır. Bit sayısı arttığında ise, her bir örneği ifade etmek için kullanılması gereken bit sayısı fazla olacağından, bu tür uygulamalar her zaman sistemin yükünü ve maliyetini arttırmaktadır. Başka bir problem ise sıklığı çok az görülen büyük genlikli işaretler için gereksiz kuanta seviyelerinin ayrılmasıdır. Bunu dışında en büyük genlikler daha küçük seviyelerde alınırsa, sinyallerde kırpmalar meydana gelecektir.

Çözüm olarak, büyük işaretler için büyük kuanta adımları, küçük işaretler için ise küçük kuanta adımları kullanılarak işaret gürültü oranının sabit tutulması sağlanabilir. Bu işlemi yapabilmek için, kullanılan haberleşme sistemlerinde, gönderici tarafında Sıkıştırma (Compressing) ve alıcı tarafında da alınan sinyal için Genleştirme (Expanding) işlemi uygulanmaktadır. Şekil 3.8'de bu sıkıştırma ve genleştirme işlemlerinin düzgün olmayan karakteristikleri görülmektedir.

Kullanılan bazı sistemlerde, sıkıştırma işlemi direk olarak analog bilgi işareti üzerinden uygulanır. Yapılan sıkıştırma işlemi sayesinde, kuanta seviyelerinin büyük kısmının kullanılması sağlanarak, her bir kuanta seviyesinden sistemin maksimum başarım alması sağlanır. Şekil 3.9'da düzgün olmayan PCM blok şeması gösterilmektedir.



Şekil 3.8 Sıkıştırma ve genleştirme eğrileri

Şekil 3.9 Düzgün olmayan kuantalama blok diyagramı

Kullanılan haberleşme sisteminde, birçok kanal kullanılıyorsa ve bu işaretler çoğaltılıp tek bir kuantalayıcı ve kodlayıcıya uygulanıyorsa, sıkıştırma işleminin her kanal için farklı bir sıkıştırıcı ile yapılması yerine, kuantalayıcı ve kodlayıcı girişinde yapılması sistem basitleştirilmesi açısından daha başarılı olur.

Düzgün olmayan sıkıştırma ve genleştirme devrelerinde genellikle logaritmik gerilim-akım ilişkisi bulunan diyotlar veya birkaç diyot ile birbirinden bağımsız değişik ağırlıklı zayıflatıcıların örneklenmiş işaretin genliğine bağlı olarak devreyi açıp kapatmasından yararlanılmaktadır. Genel olarak diyot karakteristikleri ile sağlanan bu sıkıştırma ve genişletme eğrileri birbirine tam olarak uymadıklarından, çok doğru sonuç elde etmesi istenen kuantalayıcı ve kodlayıcılarda verim sağlayamamaktadırlar.

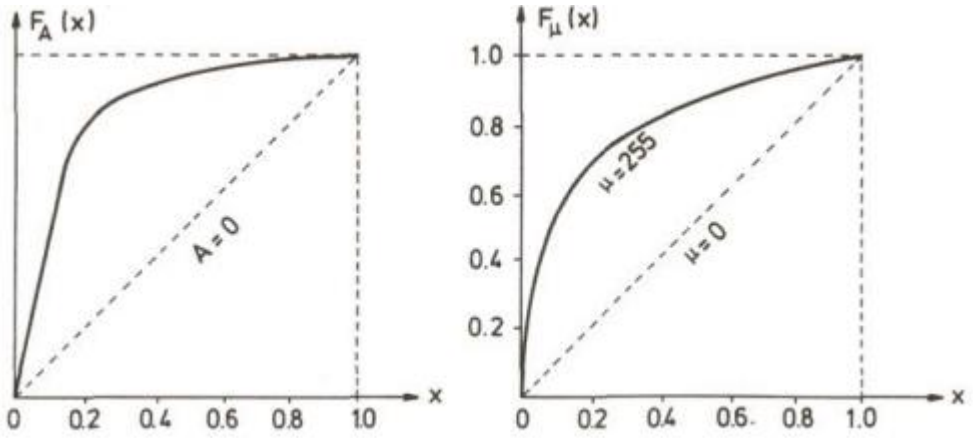
CCITT1 tarafından sayısal ses iletimi için önerilen başlıca iki tür sıkıştırma eğrisi vardır. Bu eğrilerin özellikleri sinyalin istatistiksel özelliklerine bakılarak en ideal bir biçimde kararlaştırılmaktadır. Oluşan eğriler sıfır noktasından geçmekte ve sıfır noktası civarındaki eğimi uçlardaki eğimden daha fazla olmaktadır. Bu iki eğimin oranı sıkıştırma oranı olarak adlandırılır. Bu oran arttıkça işaretin dinamiği artmaktadır.

Standartlaştırılmış iki tip sıkıştırma eğrisi vardır:

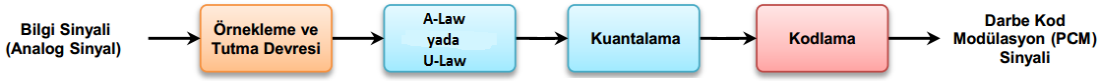
1 - Amerika ve Japonya‟da kullanılan μ-tipi (μ- Kuralı) eğri;

2 - Avrupa‟da kullanılan A-tipi (A- Kuralı) eğri;

Şekil 3.10'de μ-tipi ve A-tipi eğriler görülmektedir.



Şekil 3.10 A- kuralı ve μ– kuralı eğrileri



Şekil 3.11 PCM düzgün-olmayan kuantalama blok diyagramı

Bir bütün olarak incelediğimizde PCM Düzgün-Olmayan Kuantalama için Şekil 3.11'deki gibi bir blok diyagram elde edilir.

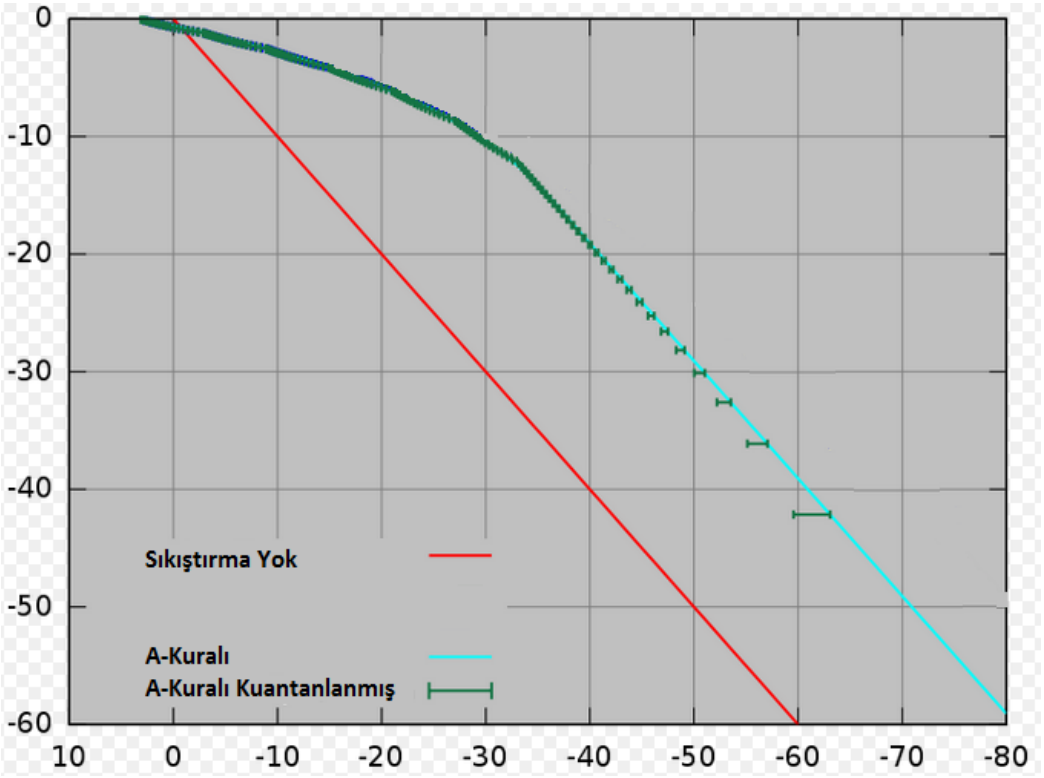
3.2.2.1. A-Kuralı

A- Kuralı algoritması standart bir sıkıştırma algoritmasıdır. Avrupa'da kullanılmakta olan 8-Bit PCM Dijital iletişim sistemlerinin analog sinyal sayısallaştırması için kullanılan standart bir yasadır (Wikipedia, 2014). Genel kullanım değeri olarak A=87.7 olarak baz alınır.

Bir x sinyali girişi için şu denklem kullanılır;

$$F_A(\mathrm{x}) = \mathrm{sgn}(\mathrm{x})\,\frac{A|x|}{1+\ln(A)} \quad , \quad |\mathrm{x}| < \frac{1}{A}$$

$$F_A(\mathrm{x}) = \mathrm{sgn}(\mathrm{x})\,\frac{1+ln(A|x|)}{1+\ln(A)} \quad , \quad \frac{1}{A} \leq |\mathrm{x}| \leq \mathrm{A}$$



Şekil 3.12 A- kuralı uygulanmış bir sinyal

Avrupa'da A değeri 87.7 olarak kullanılır ve Şekil 3.12'deki gibi bir grafik elde edilmektedir:

A- Kuralı uygulamasında kullanılan sıkıştırma fonksiyonun tersi ise alıcı kısmında kullanılır:

$$F^{-1}(y) = \operatorname{sgn}(y)\,\frac{|y|(1+\ln(A))}{A} \quad , \quad |y| < \frac{1}{1+\ln(A)}$$

$$F^{-1}(y) = \operatorname{sgn}(y)\,\frac{\exp(|y|(1+\ln(A))-1)}{A} \quad , \quad \frac{1}{1+\ln(A)} \leq |y| < 1$$

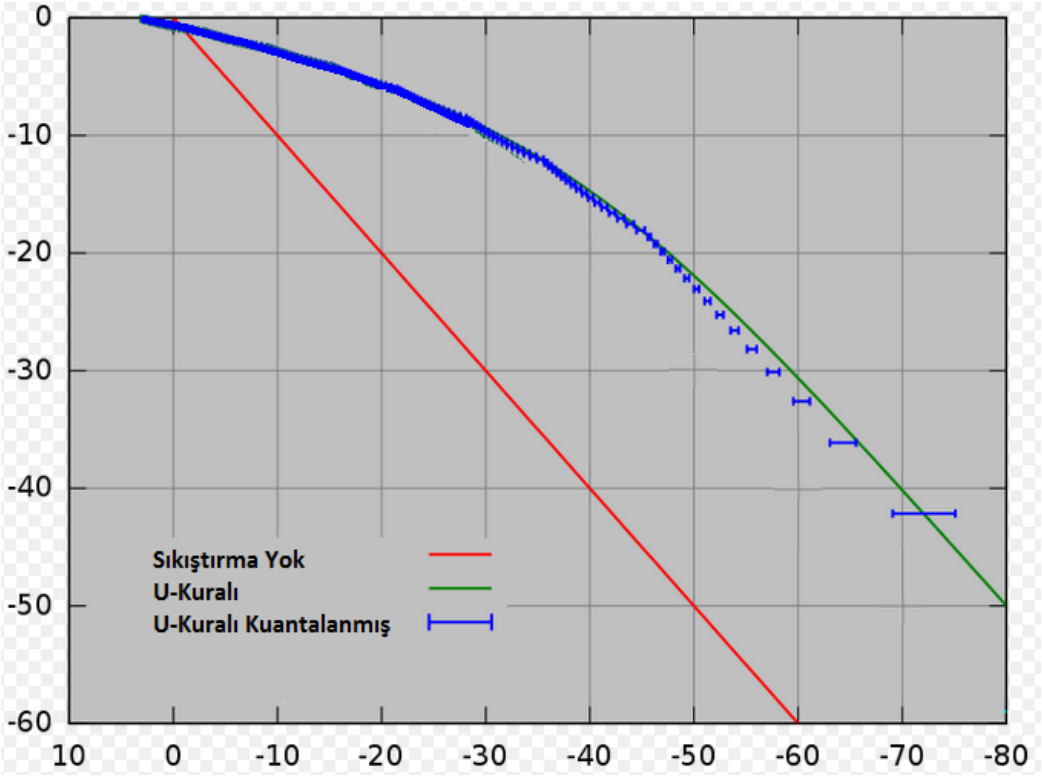
3.2.2.2. μ- Kuralı

μ - Kuralı algoritması standart bir sıkıştırma algoritmasıdır. Kuzey Amerika ve Japonya'da kullanılmakta olan 8-Bit PCM Dijital iletişim sistemlerinin analog sinyal sayısallaştırması için kullanılan standart bir yasadır (Wikipedia, 2014). Genel kullanım değeri olarak μ=255 olarak baz alınır.

Bir x sinyali girişi için şu denklem kullanılır;

$$F_{\mu}(x) = \operatorname{sgn}(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \quad , \quad -1 \leq |x| \leq 1$$

μ-kuralı için Şekil 3.13'teki gibi bir grafik elde edilmektedir:



Şekil 3.13 μ-kuralı uygulanmış bir sinyal

μ-Kuralı uygulamasında kullanılan sıkıştırma fonksiyonun tersi ise alıcı kısmında kullanılır:

$$F^{-1}(y) = \text{sgn}(y)\left(\frac{1}{\mu}\right)\left((1+\mu)^{|y|}-1\right), \quad -1 \leq |y| \leq 1$$

3.3. Kuantalama Hataları

Kuantalanmış gerçek örnek işaretlerden, kuantalanmamış gerçek örnek işaretlerin elde edilmesi hatasız mümkün değildir (Çıbuk, 2004). Yani, tersine bir işlem olmayan kuantalama sonucunda, gerçek bilginin bir kısmı yuvarlama sonucu kaybolmaktadır. Kuantalanmış örnek işaret $x_q(t)$ mesaj işareti x(t)'nin yaklaşık bir değeri olduğundan bir bozulma söz konusudur. Bu bozulmaya "kuantalama hatası" adı verilmektedir.

$$e(t) = x(t) - x_q(t)$$

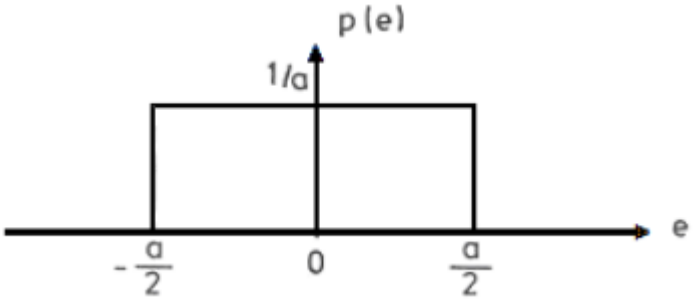
Denklemi ile gösterilen kuantalama hatasının büyüklüğü çevresel gürültünün büyüklüğüne eşdeğer kabul edilmektedir. Bu nedenle, bu bozulma "kuantalama gürültüsü" olarak adlandırılır. Oluşan gürültüyü tümüyle ortadan kaldırmak mümkün değildir ancak değişik yöntemlerle etkisi azaltılabilir.

Giriş-çıkış karakteristiği düzgün kuantalayıcı kullanılarak, kuantalama gürültüsü istatistiksel olarak hesaplanabilir. Kuantalama seviyesi (adımı) "a" olan kuantalama gürültüsünün olasılık yoğunluk fonksiyonu olur.

$$p(e) = \frac{1}{a} \quad , \quad \frac{-a}{2} < e \frac{a}{2}$$

$$p(e) = 0 \quad , \text{ diğer}$$

p(e)'nin değişimi Şekil 3.14'te gösterilmiştir.



Şekil 3.14 Kuantalama hatasının olasılık dağılımı

Kuantalama gürültüsünün karesel ortalaması:

$$<e^2> = \mathrm{E}[e^2] = \int_{\frac{-a}{2}}^{\frac{a}{2}} e^2 \mathrm{p(e)de} = \frac{a^2}{12}$$

Olarak bulunur. Ayrıca, ortalamanın E[e]= 0 olduğu görülmektedir. Şekil 3.14'de kuantalama hatasının olasılık dağılımı gösterilmektedir.

İşaretin maksimum genliği Amax ise, n-bitlik bir kodlamada, kuantalama adımı bulunabilir. Örnek olarak, en büyük gerilimin 2 volt olması durumunda 8 bit için her bir adım a=(4/8)=0.5 volt olmaktadır. Amax genliğine göre ayarlanmış n-bitlik bir kuantalayıcı genliği A olan bir sinüzoidal işarete uygulanırsa, işaretin gürültüye oranı (S/N) şöyle hesaplanabilir:

$$(\frac{S}{N}) = \frac{<x^2(t)>}{<e^2(n)>} = \frac{A^2/2}{a^2/2} = \frac{3}{2} 2^{2n} (\frac{A}{A_{\max}})^2$$

$$(\frac{S}{N})_{dB} = 10\log_{10}(\frac{S}{N}) = 1.76 + 6.02n + 20\log_{10}(\frac{A}{A_{\max}})$$

3.4. Kodlama

Belirli kesitler ile örneklenmiş sinyalin kuantalama işlemiyle kuanta seviyeleri bulunabilmektedir. Bulunan değerin iletilebilmesi için kullanılan yönteme

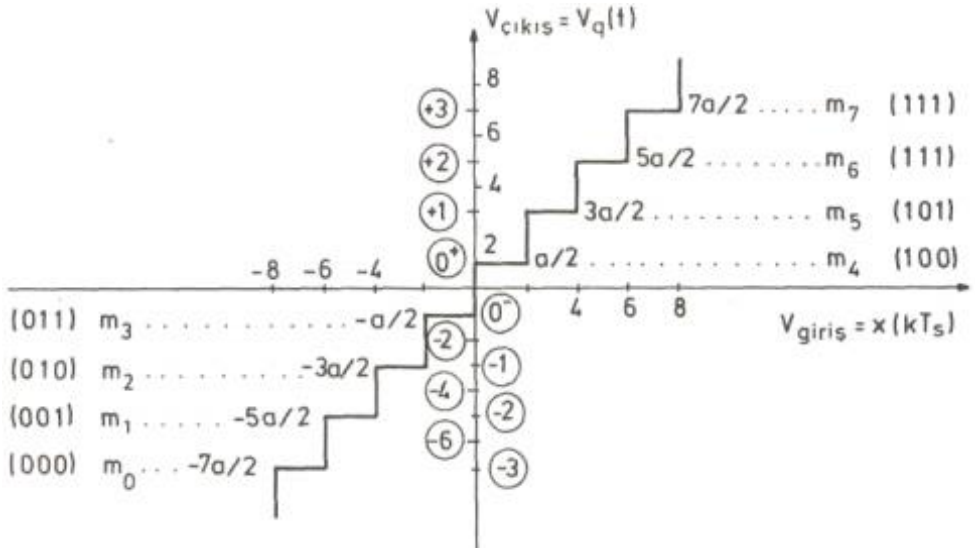
kodlama denir. Kodlama yönteminde ikili (binary) sayı sistemi kullanılır (Ertürk, 2005).

Örneklemiş bir analog sinyal, Analog Sayısal Dönüştürücüler (ADC) yardımıyla sayısal işarete dönüştürülür. Kodlama düzgün kuantalama sonucunda elde edilir.

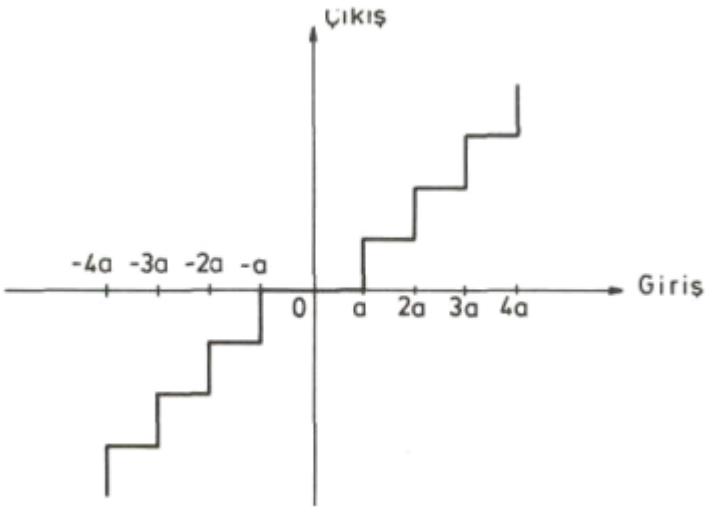
Kuantalama işleminde basamaklar 0'dan başlayarak 2^n-1'e kadar numaralandırılmaktadır. Ölçülen değerin basmak değeri ikili sayı sisteminde n basmak olacak şekilde gönderilmektedir. Şekil 3.15'te verilen eğride kuantalama kod değerleri gösterilmektedir.

3.5. Boş Kanal Gürültüsü

İşaret gürültü oranı (S/N), işaretin genliği küçüldükçe azalmaktadır. Eğer işaret, en küçük dilimden daha küçük ise, (A<a/2) gürültü işaretten daha büyük olmaktadır. Bu durum, özellikle kanal boş olduğu zaman çok rahatsız edicidir. Boş kanal gürültüsünü önlemek için, kuantalama eğrisi Şekil 3.16'daki gibi ortası yatay olacak şekilde yeniden ayarlanmaktadır. Bu yeni kuantalamada, x(t) "a" adımından küçükse daima sıfır çıkışı elde edilmektedir (Ertürk, 2005).



Şekil 3.15 Kuanta seviyeleri ve ikilik kod değerleri



Şekil 3.16 Boş kanal gürültüsünü önleyici düzgün kuanta eğrisi

4. PCM MATLAB SİMÜLASYONU

Sayısal modülasyon işlemlerinde kullanılan sinyallerin oluşturulması, örneklenmesi, kuantalanması, sıkıştırılması, kodlanması ve hata eğrilerinin analizi Matlab R2011a versiyonunda yapılmaktadır. Oluşturulan Matlab Kodları Tam Sayısal Modülasyon yöntemlerinden biri olan Darbe kod modülasyonu yöntemine uygun olarak yazılmaktadır. Oluşturulan örnek sinyaller Matlab'ta Random olarak simüle edilmekte ve grafikleri oluşturulmaktadır. Oluşturulan grafikler üzerinden uygulamada kullanılan yöntem, fonksiyon ve hata grafikleri incelenmektedir.

Yazılan kodlar incelenmesi 5 ana başlık altında yapılmaktadır:

i. Giriş sinyali (işaret)
ii. Örnekleme
iii. Sıkıştırma ve Kuantalama
iv. Kodlama (binary bitler)
v. Hata eğrilerinin incelenmesi

4.1. Giriş Sinyali (İşaret)

Giriş sinyali, programın en temel giriş kısmını oluşturmaktadır. Bu adımda üretilen sinyal daha sonra ki örnekleme, kodlama vb. işlemlerde kullanılacak olan giriş sinyaldir. Bu kısımda üretilen sinyal, 1 Hz'lik sinüs sinyali ve Matlab tarafından rastgele oluşturulan 1000 Hz'lik bir ses sinyalidir. Sinyalin genliği ise 1 ve -1 aralığında seçilmektedir. Ayrıca isteğe bağlı olarak AWGN (Additive White Gaussian Noise) Eklenebilir Beyaz Gaussian Görültüsü eklenebilmektedir. Kullanılan tüm giriş sinyallerinde çevresel etkilerin simüle edilebilmesi için AWGN kullanılmaktadır.

Giriş sinyalinin üretilmesi için aşağıdaki Matlab Kodları kullanılmaktadır.

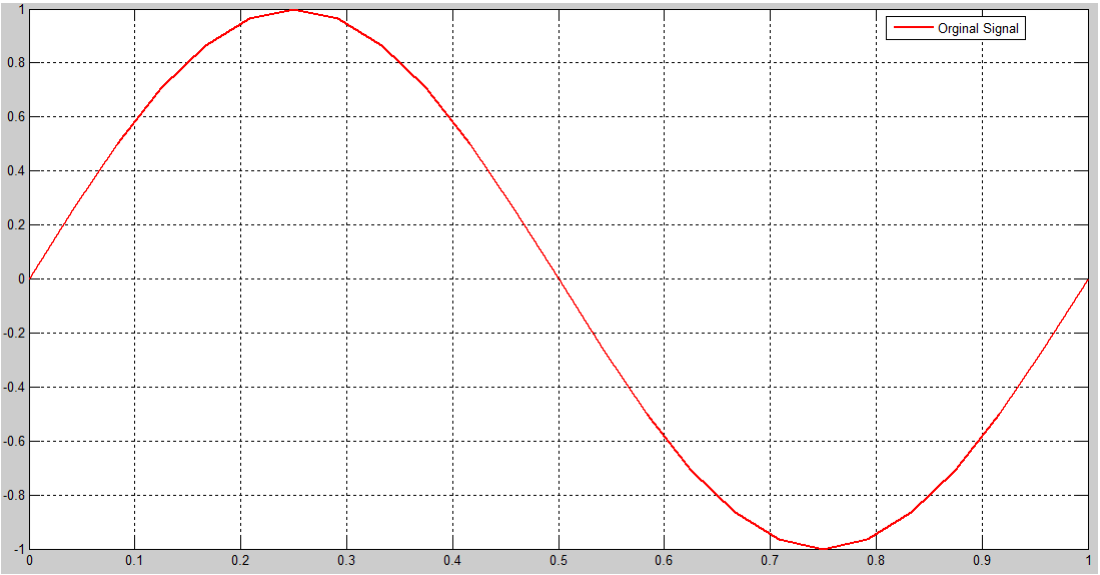
fm=1;	Kullanılan sinyalin frekansı.
A=1;	Üretilen Sinüs sinyalinin genliği.
fs=24*fm;	fs periyod başına yapılacak örnekleme sayısı.
t=0:1/(100*fm):1;	Sinyalin çizilme sıklığı.
x=A*sin(2*pi*fm*t);	Üretilen sinyal Giriş Sinüs sinyali.

Oluşturulan kodlardan üretilen sinyal Şekil 4.1 ve Şekil 4.3'te gösterilmektedir.

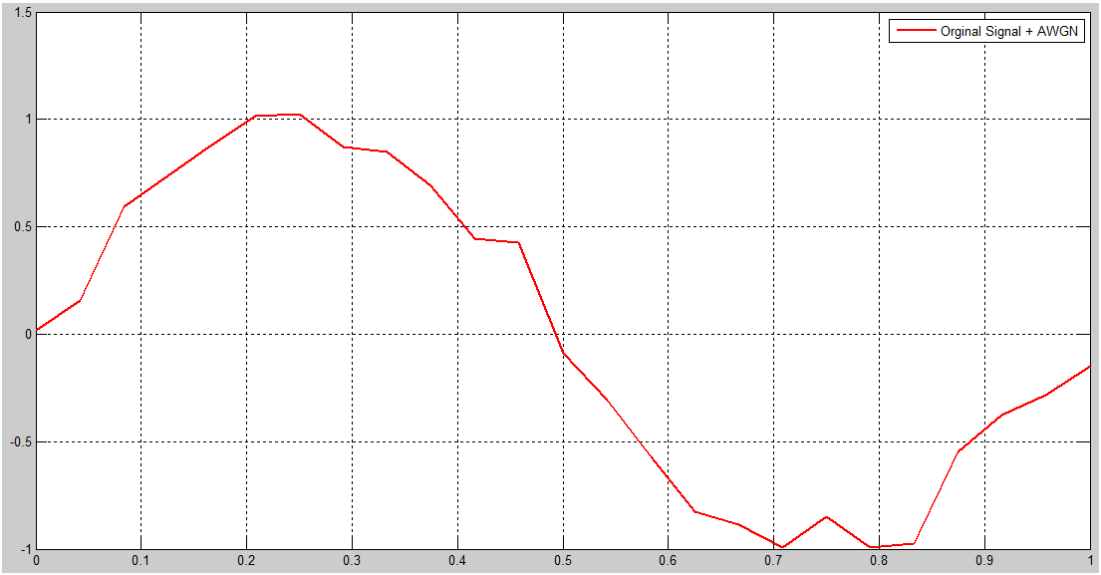
Giriş sinyaline gerçeğe daha yakın bir simülasyon olması amacı ile AWGN eklenmektedir. Bu gürültü, örnek olarak üretilen sinyale aşağıdaki Matlab Kodları ile eklenebilmektedir.

xs=awgn(xso,20); AWGN komutu giriş sinyaline Random gürültü eklemektedir.

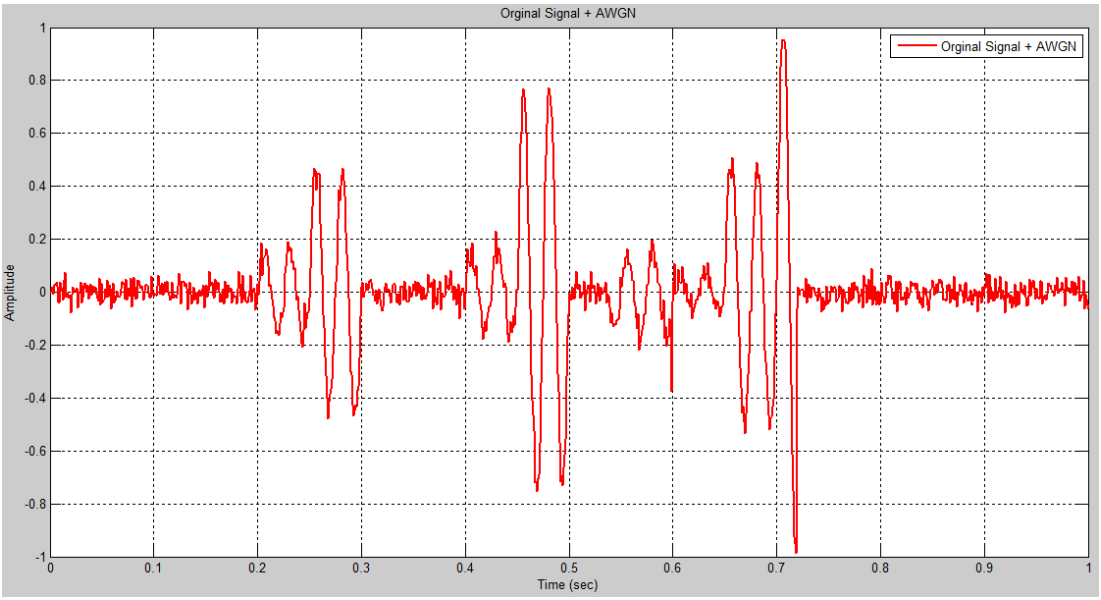
Gürültü eklenmiş giriş sinyali Şekil 4.2'de gösterilmektedir.



Şekil 4.1 Üretilen sinüs giriş sinyali



Şekil 4.2 Beyaz gürültü eklenmiş giriş sinyali



Şekil 4.3 Beyaz gürültü eklenmiş giriş ses sinyali

4.2. Örnekleme

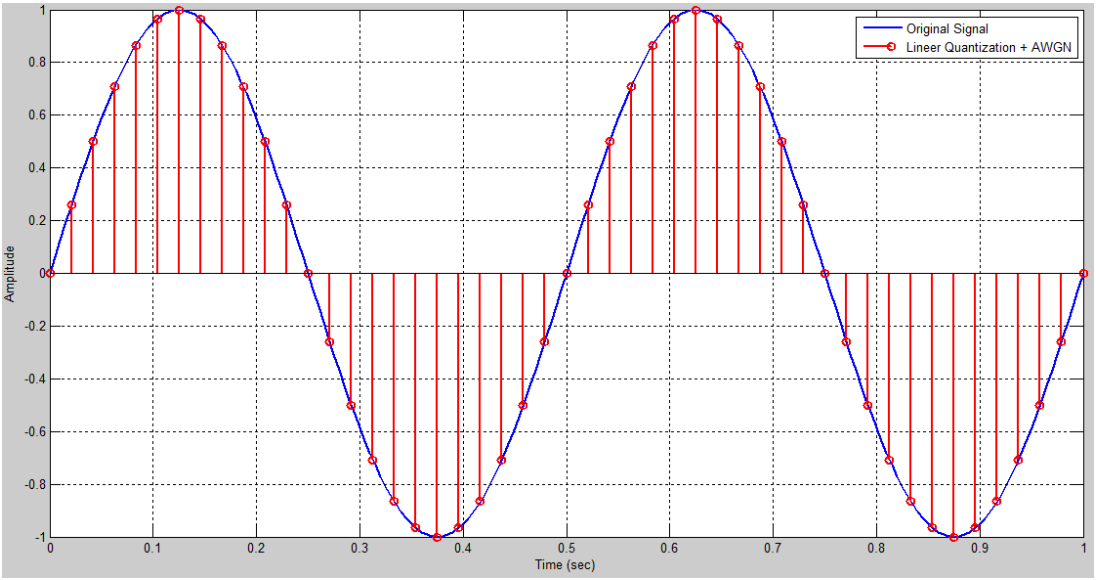
Örnekleme işlemi Matlab'ta üretilen giriş sinyalinin belirli bir örnekleme frekansı ile elde edilmesidir. Bu işlem sırasında belirlenen örnekleme sayısına göre giriş sinyalinden kesitler alınmaktadır. Kesitlerin alınması sırasında, program içinde belirtilen örnekleme sayısı ile üretilen giriş sinyalinin bir frekansından düşey olarak

eşit aralıklarda kesitler alınmaktadır. Örnekleme işleminde kullanılmakta ulan örnekleme sayısı değişken olup program başında değişmektedir. Daha sonra bu örneklenen işaretler kuantalama işleminde sayısallaştırılmaktadır.

Örnekleme işleminin gerçekleştirilmesi Matlab'ta aşağıdaki kodlar ile yapılmaktadır.

```
fm=1;
A=1;
fs=24*fm;                Örnekleme sıklığı.
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);      Daha önce oluşturulan giriş sinyali.
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);   Giriş sinyalinden oluşturulan örnek sinyal.
xs=awgn(xso,20);         Örneklenmiş sinyale gürültü eklenmesi.
```

Bu kodlar ile örneklenen sinyal, giriş sinyalinden yararlanılarak bir periyotta sinüs sinyali için 24 ve ses sinyali için 1000 adet eşit aralıklı örnekler alınarak yapılmaktadır. Sonuç olarak elde edilen grafik Şekil 4.4'te gösterilmektedir.



Şekil 4.4 Giriş sinyali ve örneklenmiş sinyal kesitleri

4.3. Sıkıştırma ve Kuantalama

Örneklenen sinyalin bir sonraki adım olarak sıkıştırılması ve kuantalanması gerekmektedir. Sıkıştırma ve Kuantalama işleminde Düzgün, A-Kuralı, μ-Kuralı ve Düzgün-Olmayan Kuantalama algoritmaları kullanılmaktadır.

Kuantalama işleminde örneklemiş sinyal işaretleri belirtilen bit sayısına göre sayısal basamaklara bölünmektedir. İstenilen bit sayısı program başında değişken olarak tanımlanmakta ve farklı değerler olarak girilebilmektedir. Örneğin; 8 bitlik bir kuantalama işlemi uygulandığında, giriş sinyalinin elde edilen maksimum genliği 2^8 = 256 eşit kuanta seviyesine ayrılmaktadır. Daha sonra elde edilen genlik değerleri karşılık düşen kuanta seviyeleri ile çarpılmaktadır. Çarpım sonucu elde edilen değer en yakın tam sayıya yuvarlanmaktadır. Elde edilen sayılar daha sonra kodlamada kullanılmaktadır.

Sıkıştırma işlemi uygulanacak sinyal, kuantalama işleminden önce belirli sıkıştırma algoritmalarına göre sıkıştırılır ve sonra düzgün kuantalama işlemi uygulanmaktadır. A-Kuralı Kuantalama işleminde optimum A değeri 87.7 olarak kabul edilmektedir. μ-Kuralı Kuantalama işleminde ise optimum μ değeri 255 olarak kabul edilmektedir.

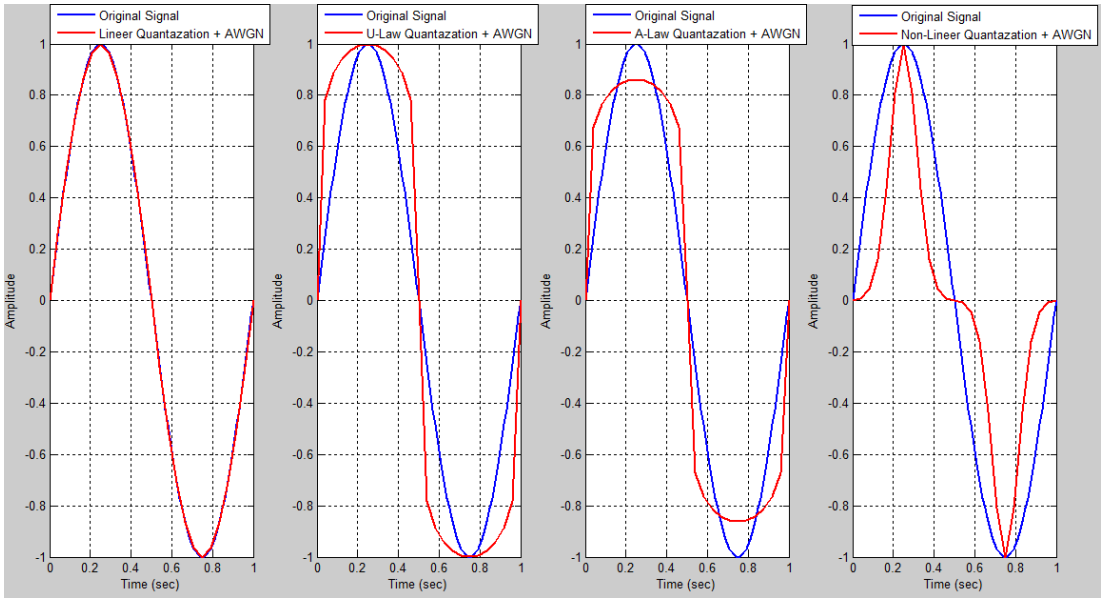
Oluşturulan Matlab Kodlarında 4 farklı Kuantalama kullanılmaktadır:

i. Düzgün Kuantalama
ii. μ-Kuralı Kuantalama
iii. A-Kuralı Kuantalama
iv. Düzgün-Olmayan Kuantalama

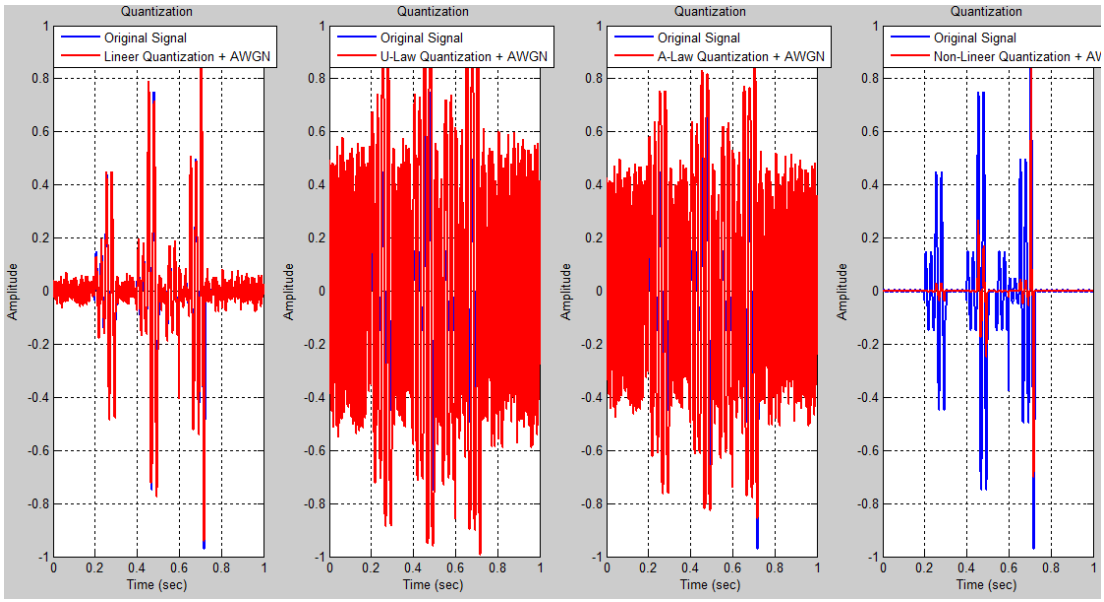
Kuantalama işlemleri Şekil 4.5 ve Şekil 4.6'da gösterilmektedir.

Kuantalama işleminden önce sıkıştırma işlemi uygulanmış ve uygulanmamış sinyaller arasında uygulanan fonksiyonlara göre farklılıklar görülmektedir. Sıkıştırma işlemi uygulanan A-Kuralı ve μ-Kuralı sinyalleri giriş sinyaline göre farklı formlara dönüşmektedir. Sıkıştırılan sinyallerin eğimlerinde azalma olarak daha doğrusal bir form aldıkları görülmektedir. Genişletme işlemi uygulanan Düzgün-Olmayan sinyal ise μ-Kuralı fonksiyonun tam zıttı karakteristik

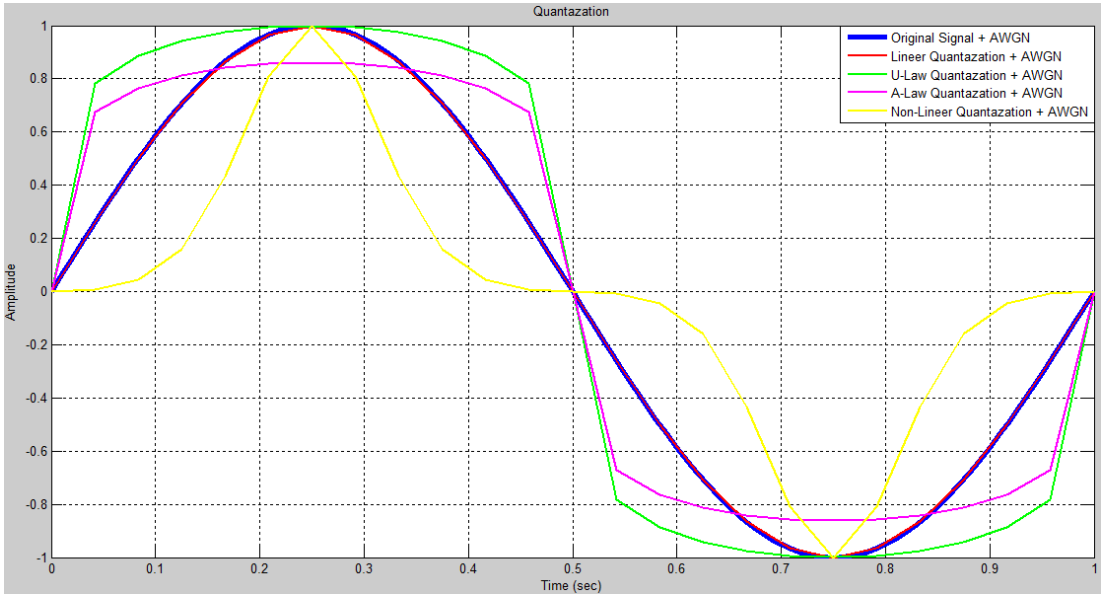
göstermektedir. Dört farklı sinyallerin aynı grafik altında incelenmesi Şekil 4.7'de gösterilmektedir.



Şekil 4.5 Sinüs sinyali için düzgün, μ-kuralı, A-kuralı ve düzgün-olmayan kuantalama



Şekil 4.6 Ses sinyali için düzgün, μ-kuralı, A-kuralı ve düzgün-olmayan kuantalama



Şekil 4.7 Sıkıştırma ve genişletme uygulan sinyallerin değişen formları

4.3.1. Düzgün Kuantalama

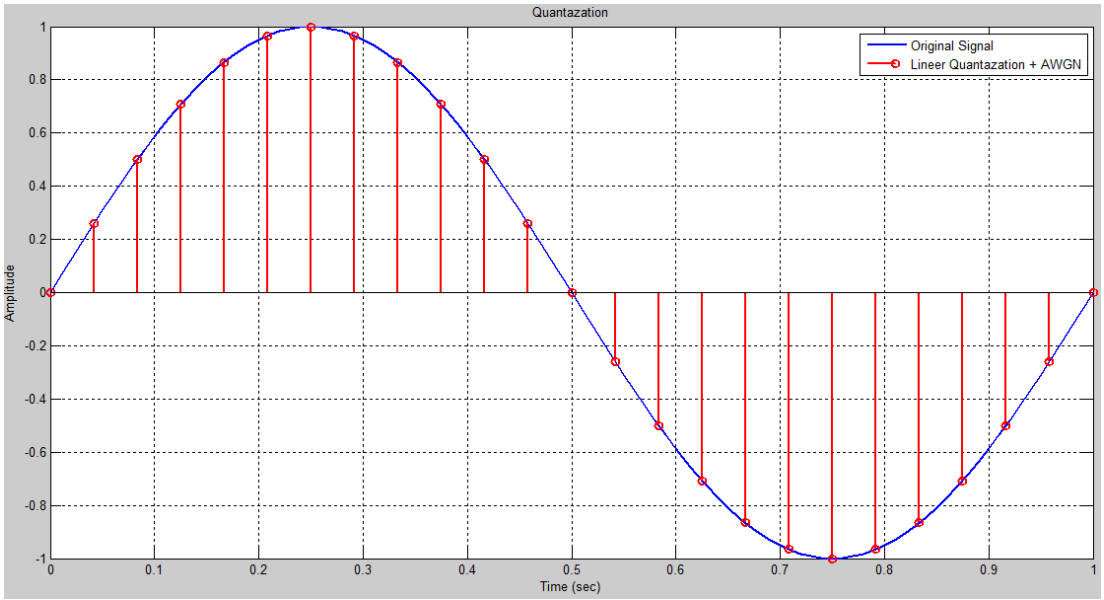
Düzgün kuantalama işlemi bit değeri 12 seçilerek, sinüs sinyali için 24 ve ses sinyali için 1000 adet örnekleme yapılarak sıkıştırma kullanılmadan elde edilmektedir. Düzgün kuantalama işleminde gürültü eklenmiş giriş sinyalinden direkt olarak örnekler alınmaktadır. Alınan örnekler eşit kuanta seviyelerine ayrıldıktan sonra tam sayı değerlerine yuvarlanarak kuantalama işlemi tamamlanmaktadır. İşlem sırasında kullanılan kodları aşağıdaki gibi incelenmektedir.

```
fm=1;
A=1;
fs=24*fm;
n=12;                          Bit sayısı.
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);         Örneklenen sinyal.
xs=awgn(xso,20);               Örneklenen sinyal + AWGN.
Mmax=max(xs);                  Max amplitude bulunuyor.
```

x1=real(xs)+Mmax;	Örneklenen sinyal + Genlik.
x1=x1/(2*Mmax);	Eşit seviyelere ayrılmaktadır.
L=(-1+2^n);	Bit seviyeleri.
x1=L*x1;	Kuanta seviyeleri bulunmaktadır.
xq1=round(x1);	En yakın tam sayıya yuvarlanmaktadır.
r1=xq1/L;	Bit sayısına bölünmektedir.
r1=2*Mmax*r1;	Sinyalin iki katı oluşturulmaktadır.
r1=r1-Mmax;	Kuantalanmış seviyeler elde edilmektedir.
y1=[];	Matrise kaydedilmektedir.

Düzgün kuantalama işlemi sonucunda elde edilen sinyal ve giriş sinyali Şekil 4.8'de gösterilmektedir.

Grafik incelendiğinde düzgün kuantalama işleminde, giriş sinyali ile kuantalanmış sinyalin formlarının aynı olduğu görülmektedir. Bir sıkıştırma işlemi uygulanmadığı için elde edilen kuantalanmış sinyal sadece kuantalama işlemi sırasında yapılan yuvarlama işleminin hatalarını taşımaktadır.



Şekil 4.8 Giriş sinyali ve düzgün kuantalanmış sinyal

4.3.1.1. μ-Kuralı Kuantalama

μ-Kuralı Kuantalama işlemi, giriş sinyaline μ-Kuralı yasasına göre uygun sıkıştırma fonksiyonun uygulanması ile yapılmaktadır. μ-Kuralı Kuantalama işleminde öncelikle giriş sinyaline, algoritmaya uygun olarak μ= 255 değerine göre sıkıştırma algoritması uygulanmaktadır. Yapılan sıkıştırma işleminden sonra elde edilen sinyal tekrar Düzgün Kuantalama kullanılarak kodlama için hazır hale getirilmektedir.

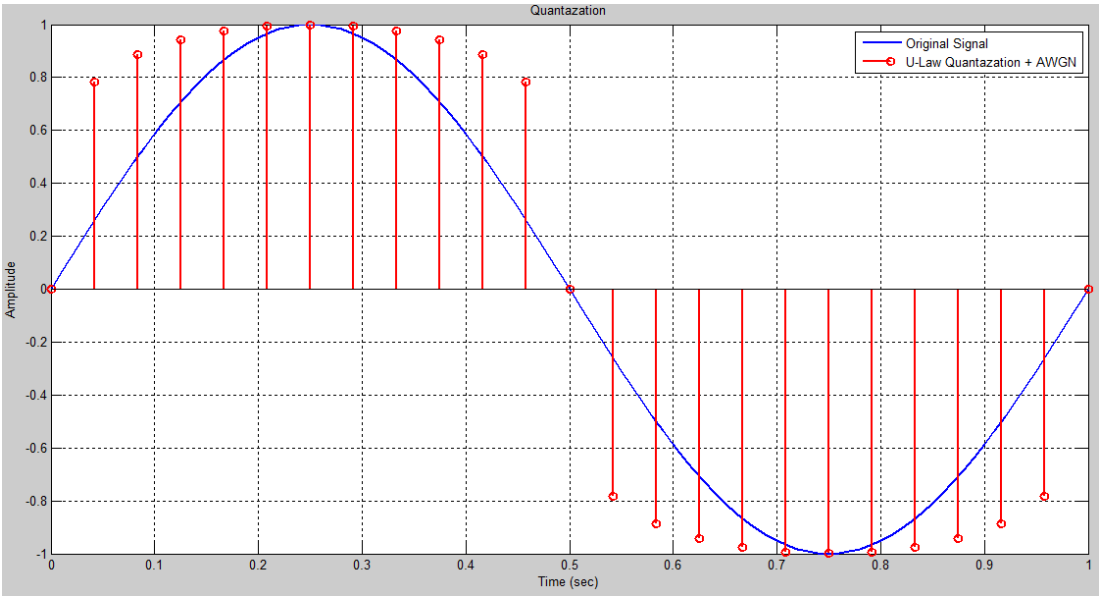
Matlab kodları aşağıdaki gibi oluşturulmaktadır.

```
fm=1;
A=1;
fs=24*fm;
n=12;                                   Bit sayısı.
u=255;                                  μ Değeri.
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);                  Örneklenen sinyal.
xs=awgn(xso,20);                        Örneklenen sinyal + AWGN.

Mn=xs/Mmax;                             Örneklenen sinyal Max genliğe
                                        bölünmektedir.
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));  μ-Kuralı fonksiyonu.

x2=real(ux)+Mmax;                       Sonraki adım Düzgün Kuantalama.
x2=x2/(2*Mmax);
L=(-1+2^n);
x2=L*x2;
xq2=round(x2);
r2=xq2/L;
r2=2*Mmax*r2;
```

```
r2=r2-Mmax;
y2=[];
```

Sıkıştırma işlemi ve kuantalama işlemi sonucunda elde edilsen sinyal Şekil 4.9'da gösterilmektedir.

Grafik incelendiğinde μ-Kuralı sıkıştırma fonksiyonu uygulanmış sinyalin alınan örneklerinin genlikleri, giriş sinyaline göre daha yüksek genlik değerlerine sahip olduğu görülmektedir. Uygulanan fonksiyon sayesinde örneklerin genliklerinin, maksimum genlik seviyesine yaklaştırıldığı ve aralarındaki farkların yani sinyalin eğiminin azaldığı görülmektedir.



Şekil 4.9 Giriş sinyali ve μ-kuralı kuantalanmış sinyal

4.3.1.2. A-Kuralı Kuantalama

A-Kuralı Kuantalama işlemi, giriş sinyaline A-Kuralı yasasına göre uygun sıkıştırma fonksiyonun uygulanması ile yapılmaktadır. A-Kuralı Kuantalama işleminde öncelikle giriş sinyaline, algoritmaya uygun olarak A= 87.7 değerine göre sıkıştırma algoritması uygulanmaktadır. Yapılan sıkıştırma işleminden sonra elde edilen sinyal tekrar Düzgün Kuantalama kullanılarak kodlama için hazır hale getirilmektedir

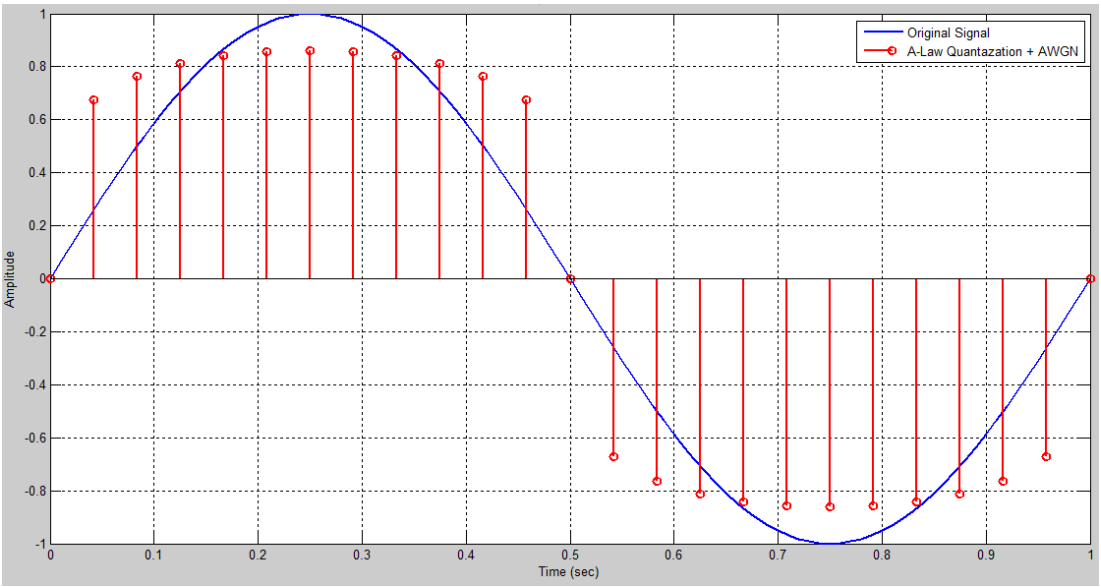
Matlab kodları aşağıdaki gibi incelenmektedir.

```
fm=1;
A=1;
fs=24*fm;
n=12;                                   Bit sayısı.
Al=87.7;                                A Değeri.
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);                  Örneklenen sinyal.
xs=awgn(xso,20);                        Örneklenen sinyal + AWGN.

if (0 <= Mn < (1/Al))                   Giriş sinyalinin 0 ve 1/A arasında olduğu
                                        kısım.
ax=Al.*Mn/(1+log(Al));                  A-Kuralı fonksiyonu.
elseif ((1/Al) <= Mn <= 1)              1/A ve 1 arasında olduğu kısım.
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al)); A-Kuralı fonksiyonu.
end

x3=real(ax)+Mmax;                       Sonraki adım Düzgün Kuantalama.
x3=x3/(2*Mmax);
L=(-1+2^n);
x3=L*x3;
```

```
xq3=round(x3);
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
```

Sıkıştırma işlemi ve kuantalama işlemi sonucunda elde edilsen sinyal Şekil 4.10'da gösterilmektedir.

Grafik incelendiğinde A-Kuralı sıkıştırma fonksiyonu uygulanmış sinyalin alınan örneklerinin genlikleri, giriş sinyaline göre, eğimin yüksek olduğu yerlerde daha büyük, maksimum genlik seviyesinde ise giriş sinyalinden daha küçük değerlere sahip olduğu görülmektedir. Uygulanan fonksiyon sayesinde örneklerin genliklerinin, ortalama ve daha doğrusal bir genlik seviyesine yaklaştırıldığı ve aralarındaki farkların yani sinyalin eğiminin azaldığı görülmektedir.



Şekil 4.10 Giriş sinyali ve A-kuralı kuantalanmış sinyal

4.3.2. Düzgün-Olmayan Kuantalama

Düzgün-Olmayan kuantalama işlemi, düzgün kuantalama işlemi uygulanmadan önce genişletme fonksiyonu kullanılarak oluşturulmaktadır. Düzgün-Olmayan kuantalama için Matlab'ta kullanılan fonksiyon μ-Kuralı fonksiyonunun tam tersi fonksiyonudur yani bir genişletme fonksiyonudur. Burada kullanılan μ değeri aynı şekilde 255 olarak kullanılmaktadır. Genişletme işlemi tamamlandıktan sonra yine düzgün kuantalama uygulanmaktadır. Daha sonra sinyal kodlama işlemi için hazır hale getirilmektedir.

Matlab kodları aşağıdaki gibi incelenmektedir.

```
fm=1;
A=1;
fs=24*fm;
n=12;                                   Bit sayısı.
u=255;                                  μ Değeri.
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);                  Örneklenen sinyal.
xs=awgn(xso,20);                        Örneklenen sinyal + AWGN.
Mmax=max(sign(xs).*(1/u).*((1+u).^abs(xs)-1));      Düzgün-Olmayan
                                                    fonksiyonu.
x4=real(sign(xs).*(1/u).*((1+u).^abs(xs)-1))+Mmax;  Düzgün-Olmayan
fonksiyonu.

x4=x4/(2*Mmax);                         Sonraki adım Düzgün Kuantalama.
L=(-1+2^n);
x4=L*x4;
xq4=round(x4);
r4=xq4/L;
r4=2*Mmax*r4;
r4=r4-Mmax;
```

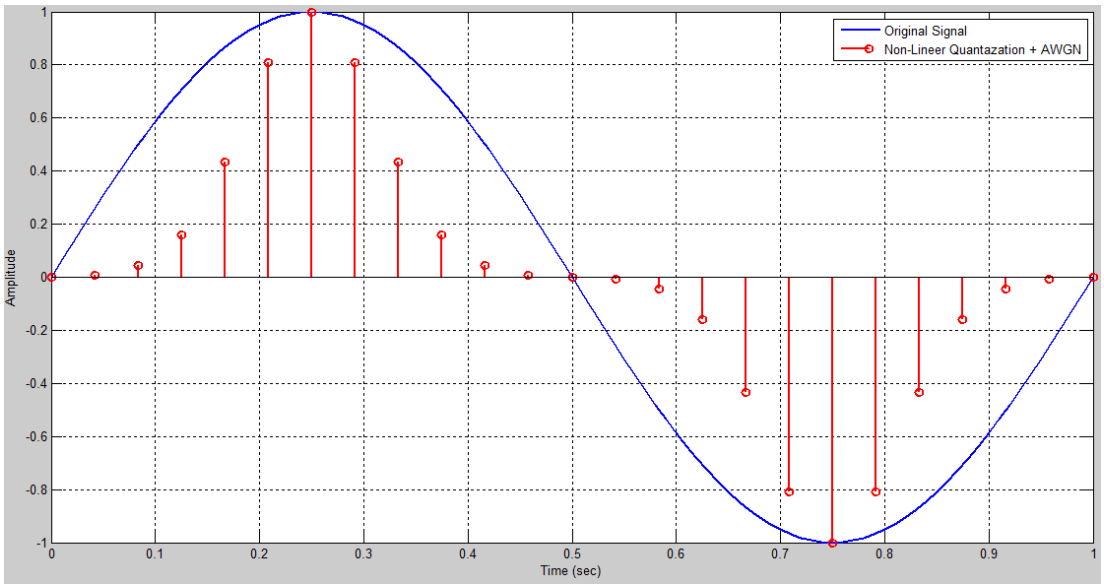
y4=[];

Genişletme işlemi ve Kuantalama işlemi sonucunda elde edilsen sinyal Şekil 4.11'de gösterilmektedir.

Grafik incelendiğinde Düzgün-Olmayan sıkıştırma fonksiyonu uygulanmış sinyalin alınan örneklerinin genlikleri, giriş sinyaline göre, eğimin yüksek olduğu yerlerde daha küçük seviyelere, maksimum genlik seviyesinde ise giriş sinyaline daha yakın değerlere yaklaştırıldığı görülmektedir. Uygulanan fonksiyon sayesinde örneklerin genliklerinin, daha küçük, eğimi daha yüksek bir genlik seviyesine yaklaştırıldığı ve aralarındaki farkların yani sinyalin eğiminin arttırıldığı görülmektedir.

4.4. Kodlama (Binary Bitler)

Kodlama işlemi, kuantalama işlemi sonrasında elde edilen sayısal örnek değerlerinin belirtilen bit sayısına göre dijital sistemler için anlaşılır ikili (binary) sayılar kümesine dönüştürme işlemidir. İkili sayılar 0 ve 1'lerden oluşmaktadır ve tüm sayısal devreler için geçerli dil olarak kabul edilmektedir.



Şekil 4.11 Giriş sinyali ve düzgün-olmayan kuantalanmış sinyal

Kuantalama işleminden sonra yazılım tarafından belirtilen bit sayısına göre örnekler ikili sayılara çevrilmektedir. Bu dönüşüm işleminde tüm kuantalama tekniklerinde aynı yöntemler kullanılmaktadır. Programda kullanılan bit sayısı n = 12 bit olarak seçilmektedir ancak isteğe bağlı değiştirilebilmektedir. Örneğin genlik değeri 100 olan bir örneğin, 12 bitlik kodlama kullanarak alacağı bit değerinin karşılığı 000001100100 olarak hesaplanmaktadır.

Matlab kodları aşağıdaki gibi incelenmektedir.

```
fm=1;
A=1;
fs=24*fm;
n=12;                          Bit sayısı.
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);         Örneklenen sinyal.
xs=awgn(xso,20);               Örneklenen sinyal + AWGN.
Mmax=max(xs);                  Max genlik bulunmaktadır.
x1=real(xs)+Mmax;              Örneklenen sinyal + Genlik.
x1=x1/(2*Mmax);                Eşit seviyelere ayrılmaktadır.
L=(-1+2^n);                    Bit seviyeleri.
x1=L*x1;                       Kuanta seviyeleri bulunmaktadır.
xq1=round(x1);                 En yakın tam sayıya yuvarlanmaktadır.
r1=xq1/L;                      Bit sayısına bölünmektedir.
r1=2*Mmax*r1;                  Sinyalin iki katı alınmaktadır.
r1=r1-Mmax;                    Kuantalanmış seviyeler elde
                               edilmektedir.
y1=[];                         Matrise kaydedilmektedir.

for i=1:length(xq1)            En büyük tamsayıya kadar döngü
                               yapılmaktadır.
d1=dec2base(xq1(i),2);         Binary bitler oluşturulmaktadır.
```

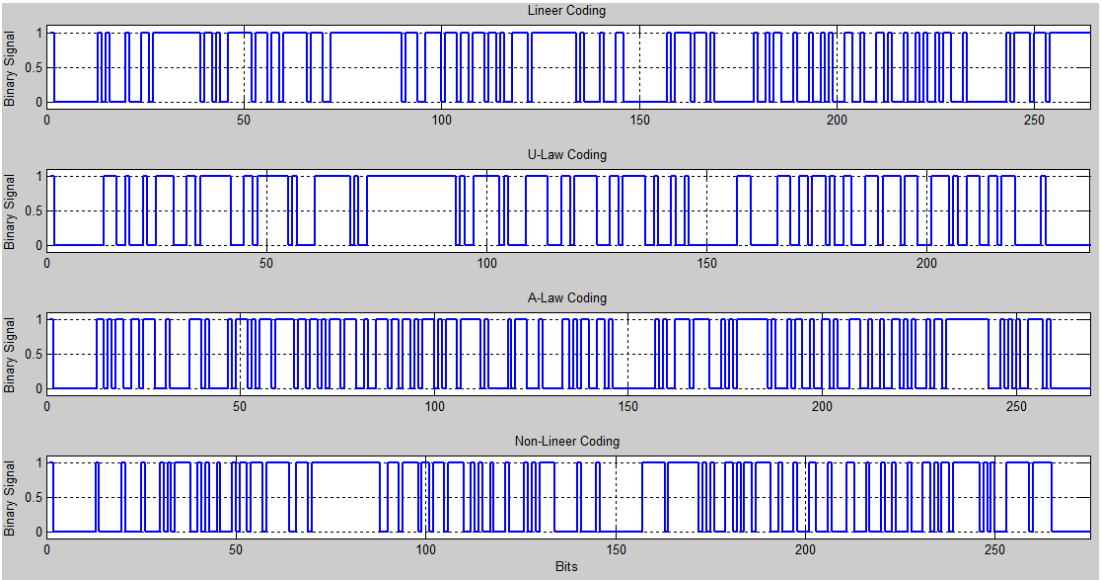
y1=[y1 double(d1)-48];	Matrise kaydedilmektedir.
end	Döngü sonu.

Bu fonksiyonlardan sonra elde edilen bitler 12'li ikili sayılar olarak gerekli matrislere kaydedilmektedir. Oluşturulan bitler Şekil 4.12 ve 4.13'te gösterilmektedir.

Grafikler incelendiğinde her kuantalama işleminde farklı bit değerleri ve sayıları elde edildiği görülmektedir. Bit sayısı olarak en düşük bit değerinin μ-Kuralı kuantalama işleminde elde edildiği görülmektedir. En yüksek bit sayısına ise Düzgün-Olmayan kuantalama işleminde ulaşıldığı görülmektedir. Bit dağılımlarının da tüm kuantalama yöntemlerinde farklı olduğu görülmektedir.

0	0	0	0	0	0	1	0	1	0
4095	4025	3821	3495	3071	2577	2048	1518	1024	600

Şekil 4.12 Binary bitler ve kuanta seviyeleri



Şekil 4.13 Kodlanan bitler

4.5. Hata Eğrilerinin İncelenmesi

Kuantalama işlemi tamamlanan sinyallerin hata grafiklerinin oluşturulması ve çizilmesi için Matlab plot ve stem fonksiyonları kullanılmaktadır. Hata grafikleri oluşturulmasında bit sayına bağlı hata grafikleri, kuantalama hata grafikleri ve kuantalama yöntemleri arasındaki hata grafiklerinin karşılaştırılması gerçekleştirilmektedir.

Grafik çizimi ile ilgili Matlab kodları aşağıdaki gibi incelenmektedir.

figure(1)	Figür numarası.
subplot(1,4,1)	1x4 alt grafik çizimi.
plot(t,x,'linewidth',2)	Sinyalin çizimi.
title('Quantazation')	Başlık.
ylabel('Amplitude')	Y eksen başlığı.
xlabel('Time (sec)')	X eksen başlığı.
hold on	Grafiği açık tutma.
stem(ts,r1,'r','linewidth',2)	Çizikler halinde 2. Sinyalin çizimi.
legend('Original Signal','Lineer Quantazation + AWGN');	Grafik yazısı.
grid	Karesel ızgara.

Hata grafikleri bit sayılarına, A değerine, μ değerine, kuanta seviyelerine göre çeşitli bölümler altında incelenmektedir.

4.5.1. Kuantalama Hataları

Kuantalama hataları, kuantalama işlemi yapılırken kullanılan bit sayısı ve örnekleme sayısına bağlı olarak ortaya çıkan hatalardır. Kuantalama işleminde kullanılan bit sayısı ne kadar yüksek olursa hata oranları da o derecede azalmaktadır. Hatalar, kuanta seviyelerinin tam sayılara yuvarlanırken ve örneklerin sıklıkları alınırken yapılan yuvarlama işleminden kaynaklanmaktadır. Şekil 4.14'te 4 farklı

kuantalama işlemi için sinüs ve ses sinyali arasındaki ortalama hata seviyelerinin karşılaştırmasını içermektedir.

Sinüs sinyali grafiği incelendiğinde tüm ortalama hata eğrileri, bit sayıları 12 ve örnekleme sıklığı 24 olarak alındığı için kuantalama hataları birbirine çok yakın bulunmaktadır. Ancak ortalama hata seviyesi olarak en düşük ortalama hata değeri A-kuralı kuantalama yönteminde % 6,6 x 10^{-3} olarak elde edilmektedir. En yüksek ortalama hatanın ise % 7,44 x 10^{-3} ile düzgün kuantalama işleminde görülmektedir.

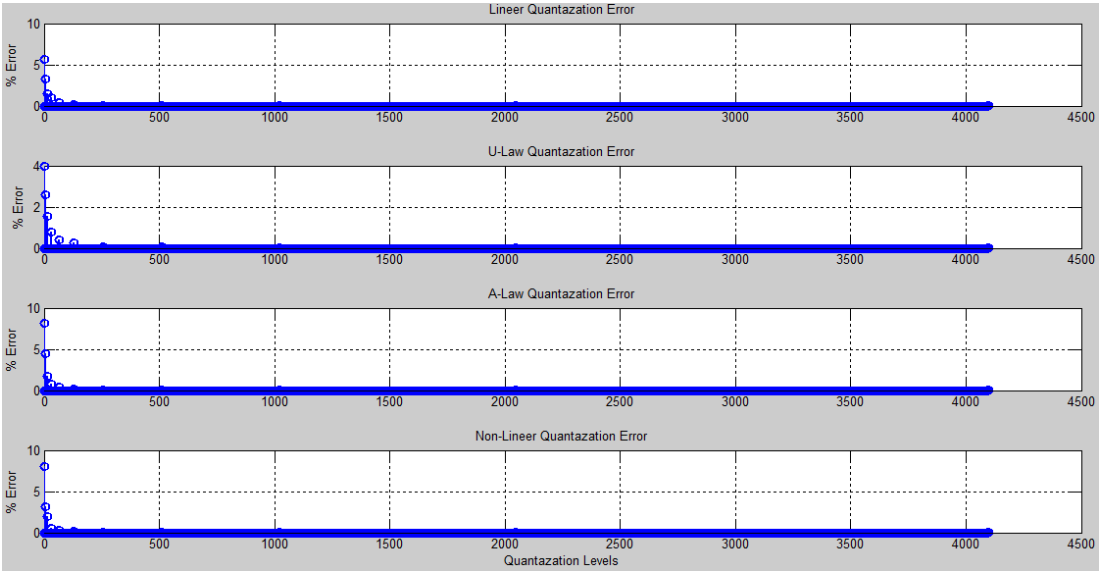
Ses sinyali grafiği incelendiğinde tüm ortalama hata eğrileri, bit sayıları 12 ve örnekleme sıklığı 1000 olarak alındığı için kuantalama hataları birbirine çok yakın bulunmaktadır. Ancak ortalama hata seviyesi olarak en düşük ortalama hata değeri μ-kuralı kuantalama yönteminde % 6,1 x 10^{-3} olarak elde edilmektedir. En yüksek ortalama hatanın ise % 7,5 x 10^{-3} ile düzgün kuantalama işleminde görülmektedir.

Aynı bit sayıları ve örnekleme sayıları kullanarak bulunan ortalama hata değerleri, sıkıştırma işlemi uygulanan kuantalama yöntemlerinde yapılan kuantalama hatalarından daha düşük olduğunu göstermektedir ve farklı iki giriş sinyali kullanıldığında da sonuçların çok fazla değişmediği görülmektedir.

Değişken bit sayısı kullanıldığında, yani değişken kuanta seviyeleri kullanıldığında ise yapılan hata grafikleri Şekil 4.15'te gösterilmektedir.

	Doğrusal Kuantalama	μ-Kuralı Kuantalama	A-Kuralı Kuantalama	Doğrusal-Olmayan Kuantalama	
Sinüs sinyali için kuantalama yöntemlerine bağlı ortalama hataların karşılaştırılması	% 7,44 x 10^{-3}	% 7,3 x 10^{-3}	% 6,6 x 10^{-3}	% 7,1 x 10^{-3}	% Hata
Ses sinyali için kuantalama yöntemlerine bağlı ortalama hataların karşılaştırılması	% 7,5 x 10^{-3}	% 6,1 x 10^{-3}	% 6,4 x 10^{-3}	% 6,5 x 10^{-3}	

Şekil 4.14 Kuantalama yöntemlerine bağlı ortalama hataların karşılaştırılması



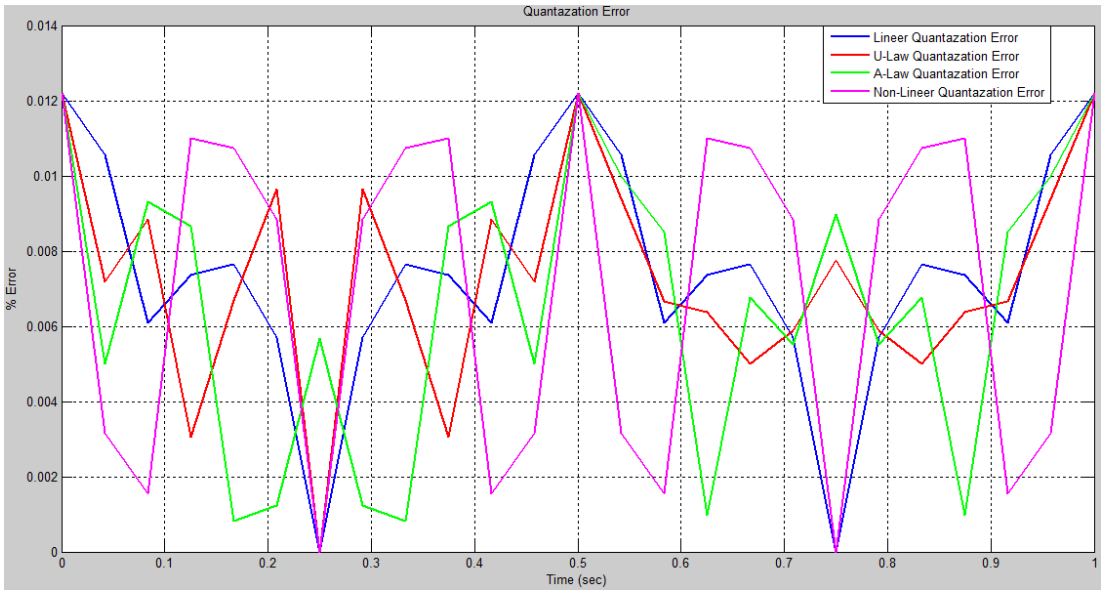
Şekil 4.15 Değişken kuanta seviyelerine bağlı hata grafikleri

Kuanta seviyeleri 2 - 4046 arasında kullanıldığında düşük kuanta seviyelerindeki hataların daha yüksek, yüksek kuanta seviyelerindeki hataların ise daha düşük olduğu görülmektedir. Genel olarak hata grafikleri incelendiğinde ise en başarılı kuantalama tekniği, en düşük hata değeri ile μ-kuralı kuantalama olarak görülmektedir. Düşük kuanta seviyelerinde ise en yüksek hata değerleri A-kuralı ve düzgün-olmayan kuantalama yönteminde görülmektedir.

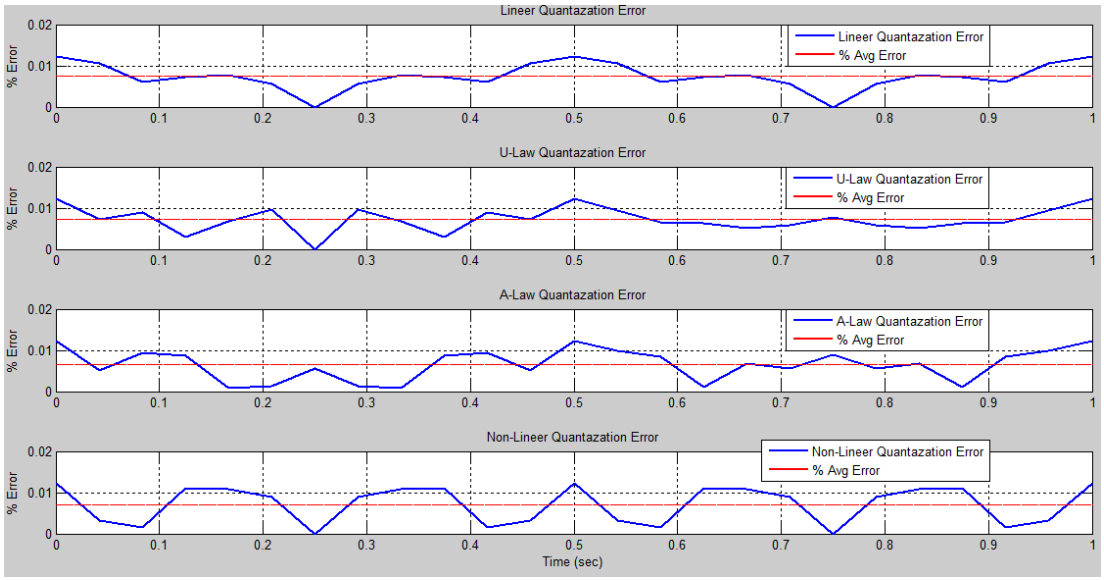
Sinüs sinyali için bir periyod içinde alınan 24 örneğin hata seviyeleri Şekil 4.16'da gösterilmektedir.

Bit sayısının 12 ve 24 örneğin kullanıldığı grafikte en az dalgalanma yani örnekler arasında ki hata farklarının değişimi A-Kuralı Kuantalama yönteminde görülmektedir. En büyük dalgalanma ise Düzgün Kuantalama yönteminde görülmektedir. Genel olarak tüm teknikler için yapılan hataların en büyük olduğu seviyeler ise 0 değerine yakın yerlerde alınan örneklerden geldiği grafikten görülmektedir. Bu hatanın sebebi ise boş kanal gürültüsünün oluşmaması için 0'a yakın değerlerin 0 seviyesine yuvarlamasından kaynaklanmaktadır.

Tüm kuantalama yöntemleri için oluşturulan karşılaştırma grafikleri Şekil 4.17'de ayrı ayrı gösterilmektedir.



Şekil 4.16 24 örneğin anlık hata değerleri

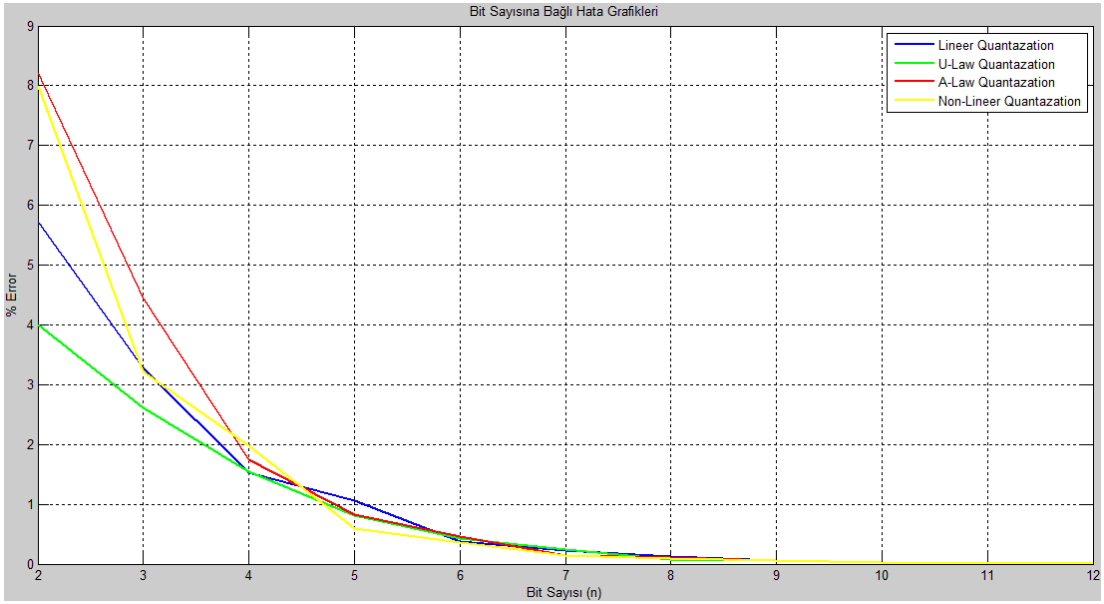


Şekil 4.17 Kuantalama yöntemleri anlık ve ortalama hata grafikleri

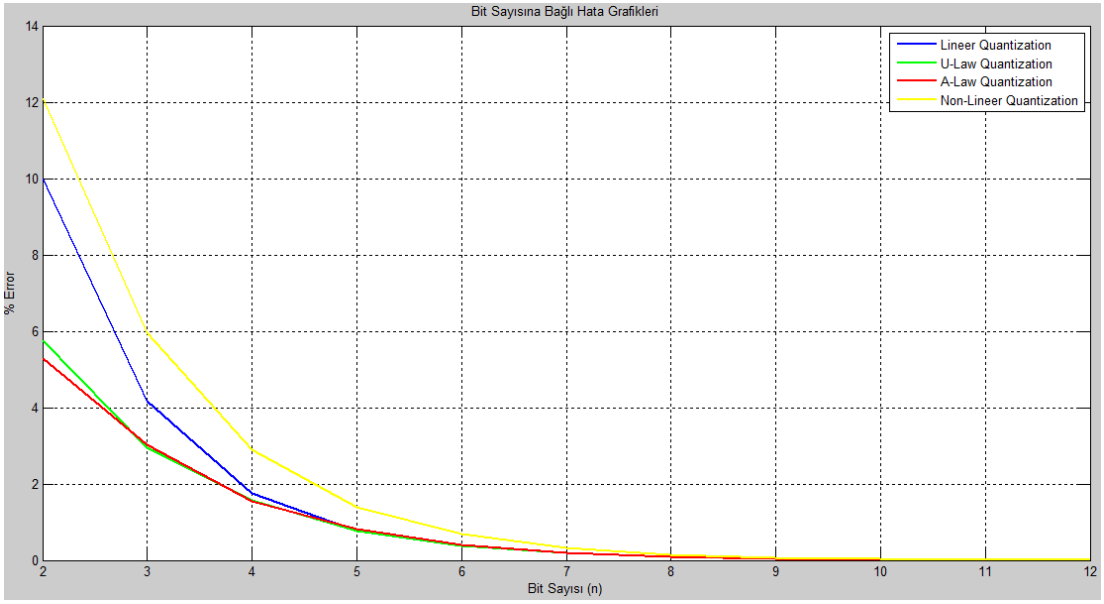
4.5.2. Bit Sayısına Bağlı Hata Grafikleri

Bit seviyesine bağlı hata grafiklerindeki hata oranları direk olarak bit sayısı ile orantılıdır. Sinüs sinyali için 24 ve ses sinyali için 1000 örnekleme sayısı, sabit A ve μ değeri kullanılarak bit sayısı arttığında kuantalama seviyeleri arttığı için genlik seviyesinde alınan kuanta sayısı artmaktadır. Genlik seviyesinde, artan örnek

sayısının tam sayıya yuvarlaması yapılırken elde edilen hatalar aza indirgemektedir. 1 – 12 bit aralığında çizdirilen 4 farklı kuantalama yöntemine ait grafikler Şekil 4.18 ve Şekil 4.19'da gösterilmektedir.



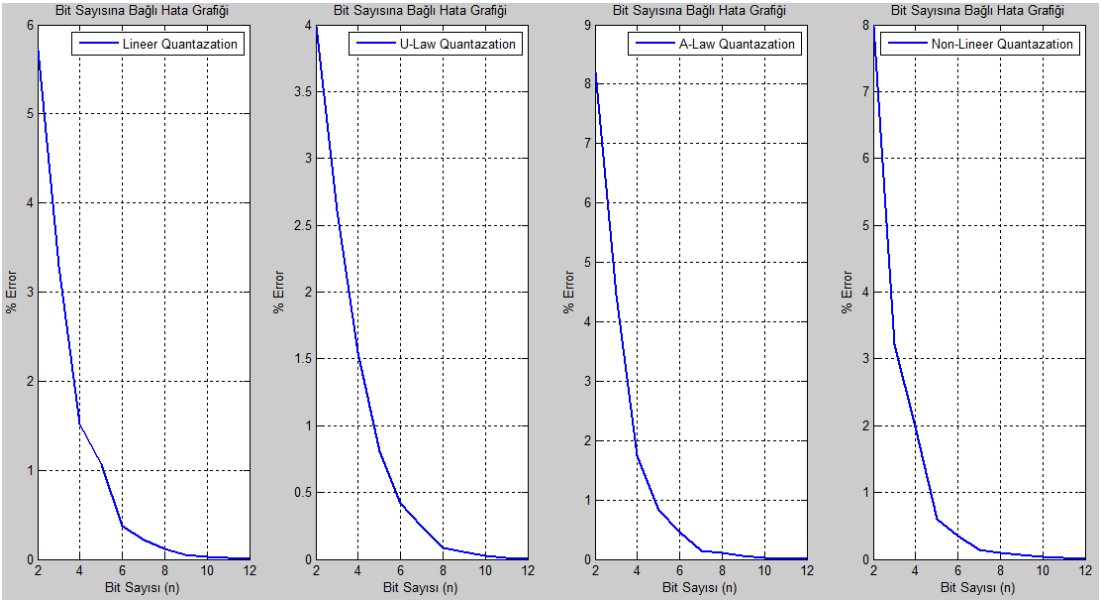
Şekil 4.18 Sinüs sinyali için değişken bit sayısına bağlı kuanta hataları



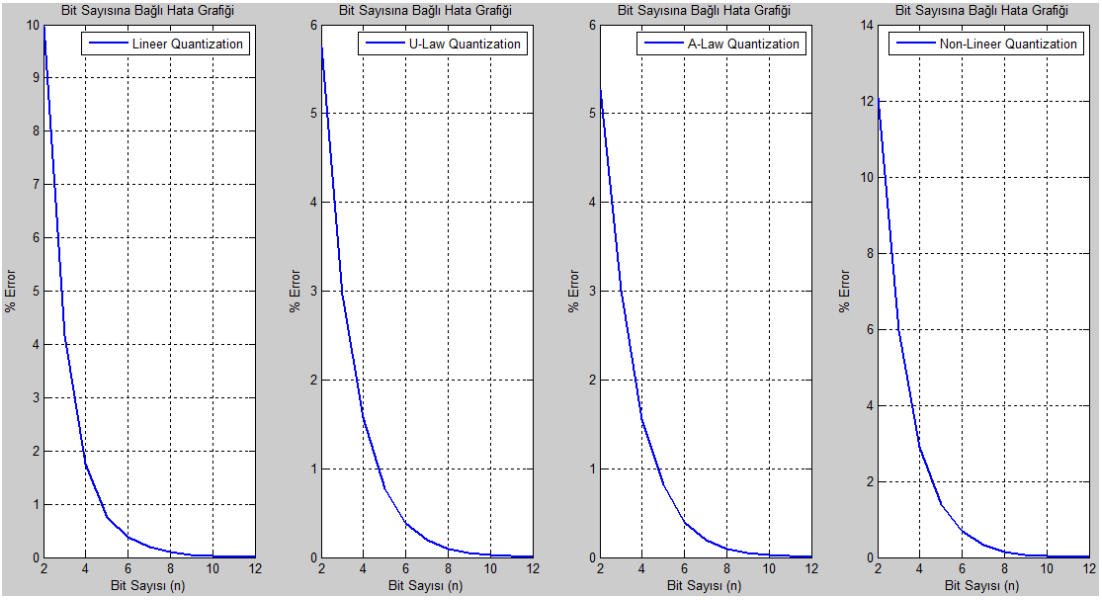
Şekil 4.19 Ses sinyali için değişken bit sayısına bağlı kuanta hataları

Grafik incelendiğinde, sinüs sinyali için az bit sayısı kullanılarak yapılan kuantalama işlemlerinde hataların en fazla olduğu yöntem A-kuralı, en düşük hatanın olduğu yöntem ise μ-kuralı olduğu görülmektedir. Ses sinyalinde ise az bit sayısı kullanılarak yapılan kuantalama işlemlerinde hataların en fazla olduğu yöntem Düzgün-Olmayan kuantalama, en düşük hatanın olduğu yöntem ise A-kuralı olduğu görülmektedir. Sinüs sinyali için başlangıçta daha yüksek hata oranına sahip A-kuralı 4 bitten sonraki seviyeler için daha iyi sonuçlar elde etmektedir ve μ-kuralı ile beraber yüksek bit seviyelerinde en düşük hata oranına sahiptir. Ses sinyalinde ise düşük bit seviyelerinde başarılı olan μ-kuralı ve A-kuralı yüksek bit seviyelerinde de düşük hata değerlerine sahiptir. İki farklı giriş sinyali içinde ortak olarak elde edeceğimiz sonuç, sıkıştırma işlemi kullanılan kuantalama yöntemlerinin optimum seviye olan 8 bit değerinde daha başarılı olmasıdır.

Tüm hata grafikleri ayrı ayrı incelendiğinde ise ortak olarak görülen; bit sayısı arttırıldığında hataların azaldığı ve 10 bit sayısına ulaşıldıktan sonra yapılan hataların çok düşük değerlere ulaştığıdır. Bit sayısına bağlı hata grafikleri Şekil 4.20 ve Şekil 4.21'de gösterilmektedir.



Şekil 4.20 Sinüs sinyali için bit seviyelerine bağlı hata grafikleri



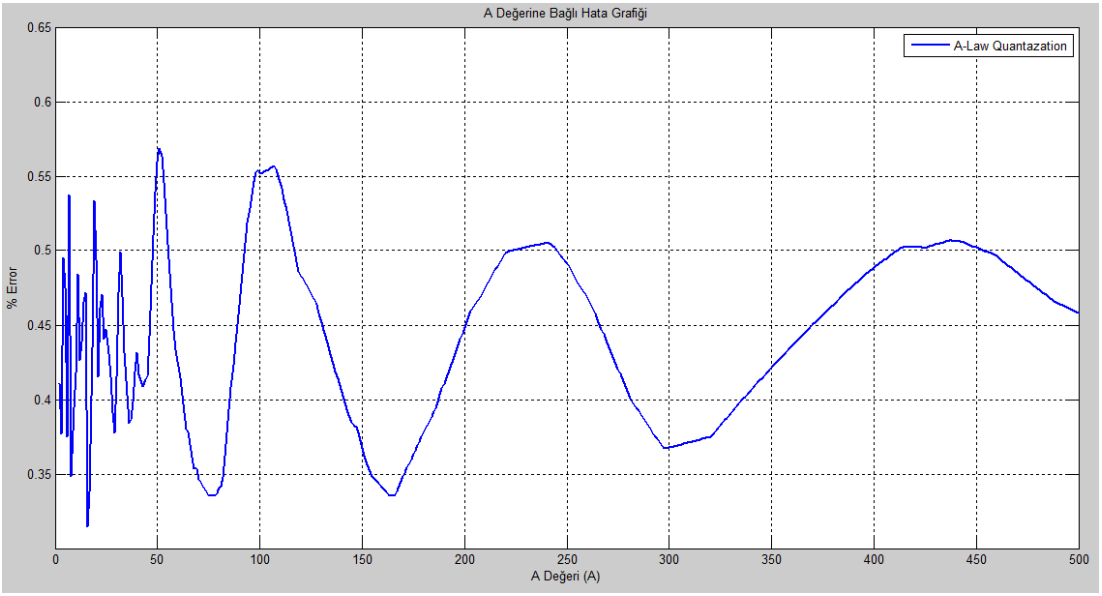
Şekil 4.21 Ses sinyali için bit seviyelerine bağlı hata grafikleri

Hata grafikleri incelendiğinde düşük bit değerlerinde en başarılı kuantalama yöntemi μ-Kuralı olarak görünmektedir. Ancak bit seviyeleri arttırıldığında, yüksek bit sayılarındaki hata oranında en başarılı A-Kuralı kuantalama yöntemi olarak görülmektedir.

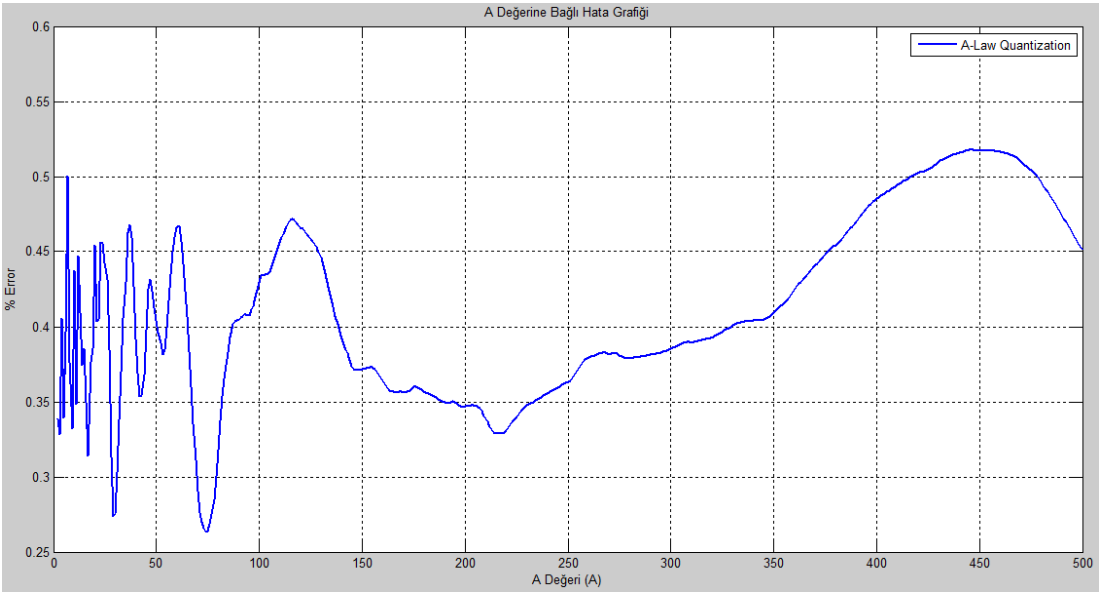
4.5.3. Kuantalama Yöntemlerine Bağlı Hata Grafikleri

Değişen kuantalama yöntemleri, değişen A ve μ değerleri kullanıldığında çizilen grafikler farklılık göstermektedir. Bu grafiklerdeki bit sayıları ve örnekleme sayıları sabit tutulup A ve μ değerleri değiştirilmektedir. Bit sayısı olarak 12 bit, örnekleme sayısı olarak sinüs sinyali için 24 ve ses sinyali için 1000 kullanılmaktadır.

İlk olarak A değerinin değişimine bağlı hata grafikleri Şekil 4.22 ve Şekil 4.23'te gösterilmektedir.



Şekil 4.22 Sinüs sinyali için değişken A değerine bağlı hata grafiği

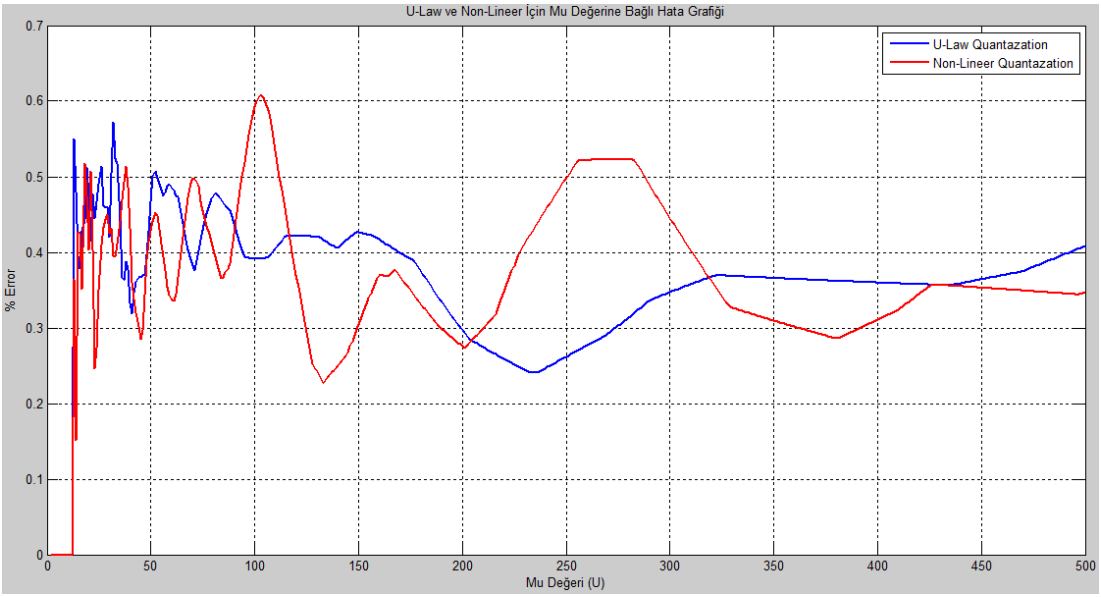


Şekil 4.23 Ses sinyali için değişken A değerine bağlı hata grafiği

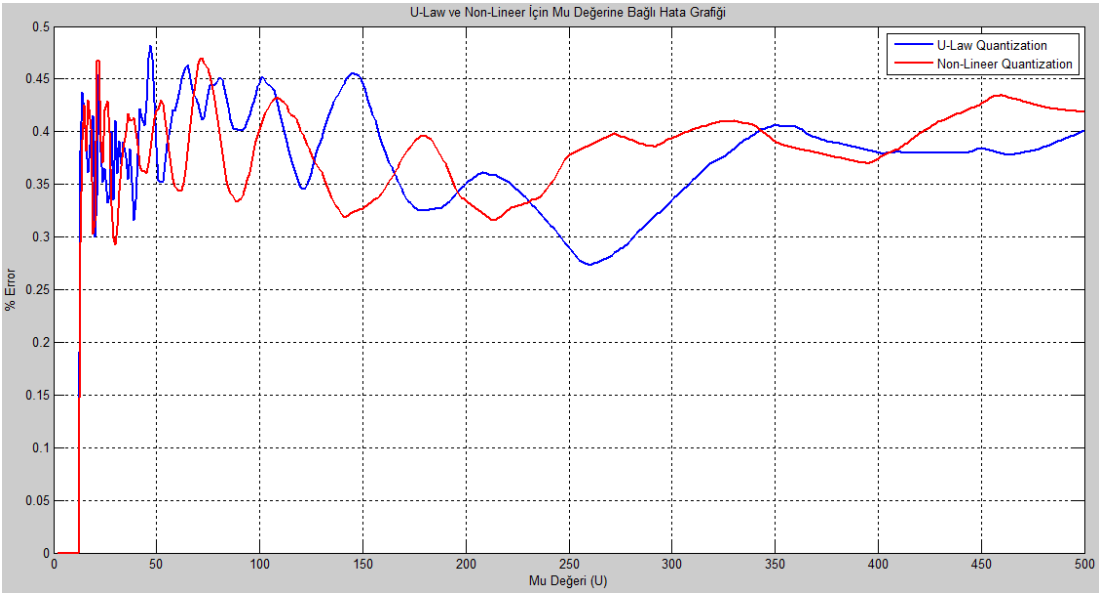
Grafikler incelendiğinde en başarılı seviye olarak, A değerinin 80 değerleri civarında ki hataların en düşük olduğu görülmektedir. Diğer A değerlerinde yapılan hataların ise daha yüksek olduğu görülmektedir. Optimum olarak kabul edilen A değeri 87.7 olduğu yaklaşık olarak grafikten de görülmektedir. Ancak 12 bit kullanılan bu grafikte, en başarılı seviye A = 81 değeri kullanıldığında elde

edilmektedir. Ayrıca grafiklerde A değeri 50 sayısına ulaşana kadar yapılan hataların çok düzensiz olduğu görülmektedir.

İkinci olarak değişken μ değerine bağlı μ-kuralı kuantalama ve ters fonksiyonu olan düzgün-olmayan kuantalama yöntemi Şekil 4.24 ve Şekil 4.25'te incelenmektedir.



Şekil 4.24 Sinüs sinyali için değişken μ değeri İçin μ-kuralı ve düzgün-olmayan kunatalama hataları

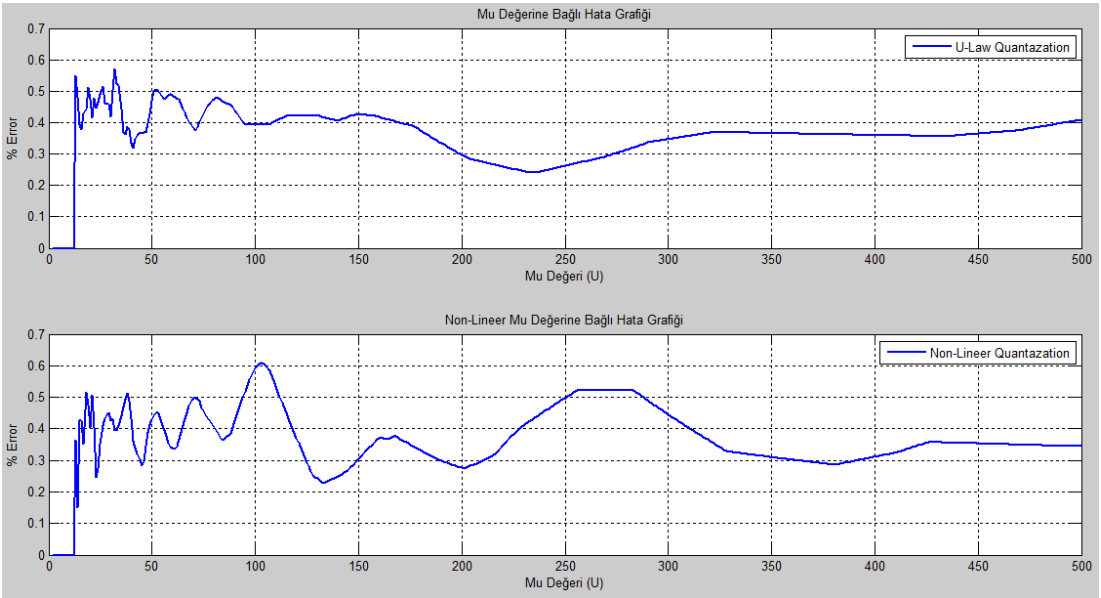


Şekil 4.25 Ses sinyali için değişken μ değeri İçin μ-kuralı ve düzgün-olmayan kunatalama hataları

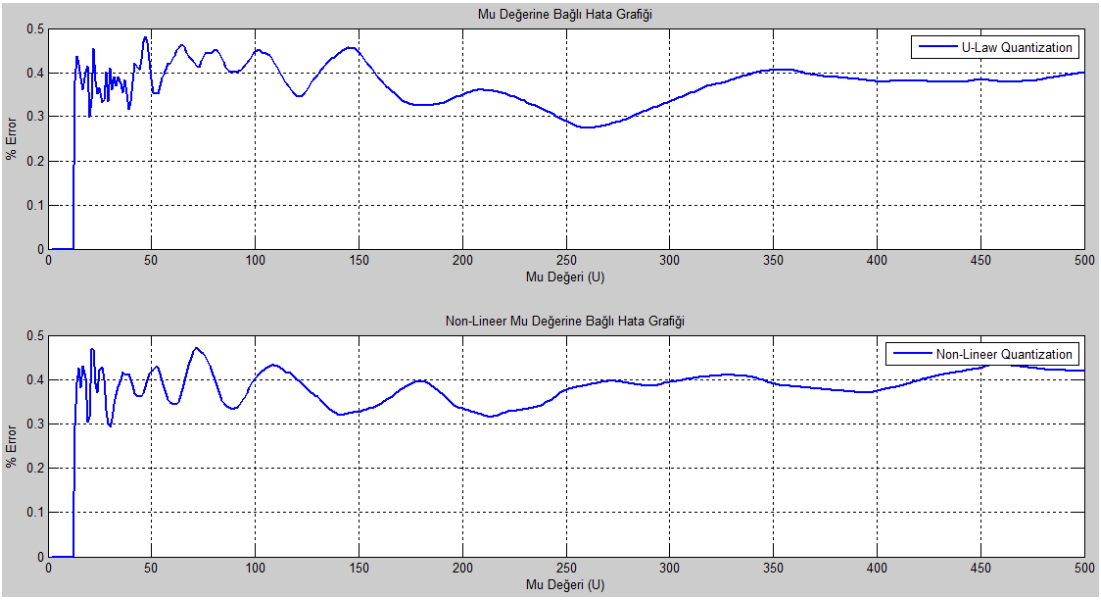
Değişen μ değeri kullanıldığında, μ-Kuralı ve Düzgün-Olmayan Kuantalama yönteminde μ = 90 değerine kadar yapılan hataların çok düzensiz olduğu görülmektedir. μ-Kuralı için en iyi hata, sinüs sinyali için μ = 240 değerinde, ses sinyali için μ = 255 değerinde elde edilmektedir. μ-kuralı sıkıştırma yöntemi küçük genliklerin çok, büyük genliklerin ise az görüldüğü sinyaller için yapılmış bir sıkıştırma yöntemidir. Bu forma uyan bir giriş sinyali verildiğinde, örnek olarak kullandığımız ses sinyali gibi bir sinyal kullanıldığında en iyi hata oranının μ = 255 değerinde olduğu görülmektedir. Bu da μ-kuralının, μ = 255 seviyesinde optimum olduğunu göstermektedir.

Düzgün-Olmayan Kuantalama yönteminde ise en iyi hata μ = 135 ve μ = 220 değerinde elde edilmektedir.

İki yöntem ayrı olarak ele alındığında ise μ = 150 değerinden sonra μ-Kuralı hata grafiğinin düzensizliği azaldığı ve düzlemsel bir forma yaklaştığı görülmektedir. Düzgün-olmayan kuantalama yönteminde ise tam olarak düzlemsel bir form elde edilememektedir. Grafikler Şekil 4.26 ve Şekil 4.27'de gösterilmektedir.



Şekil 4.26 Sinüs sinyali için değişken μ değerine bağlı hata grafikleri

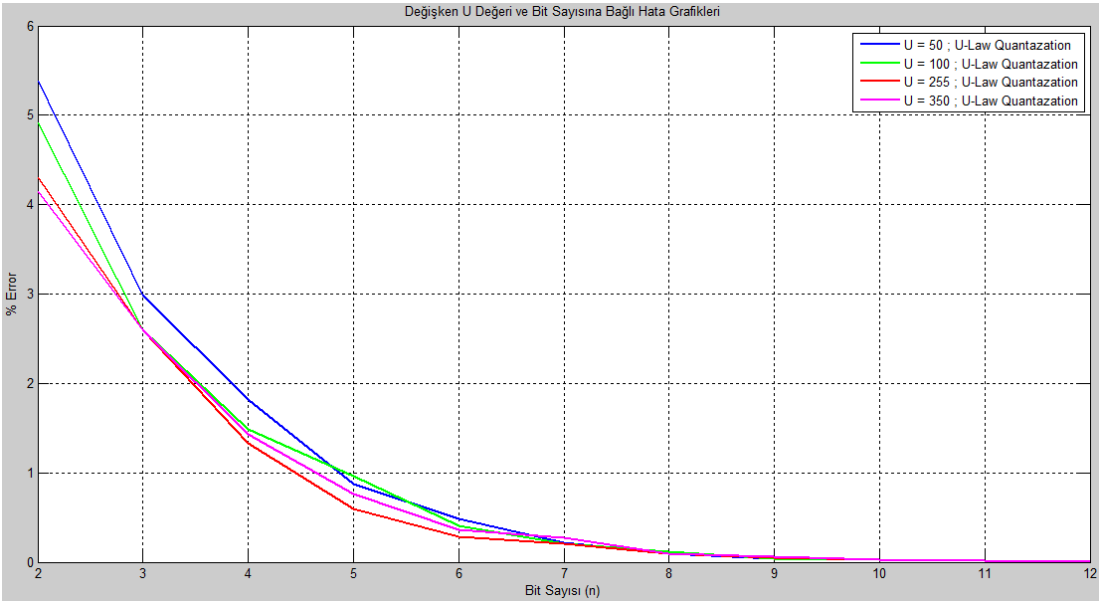


Şekil 4.27 Ses sinyali için değişken μ değerine bağlı hata grafikleri

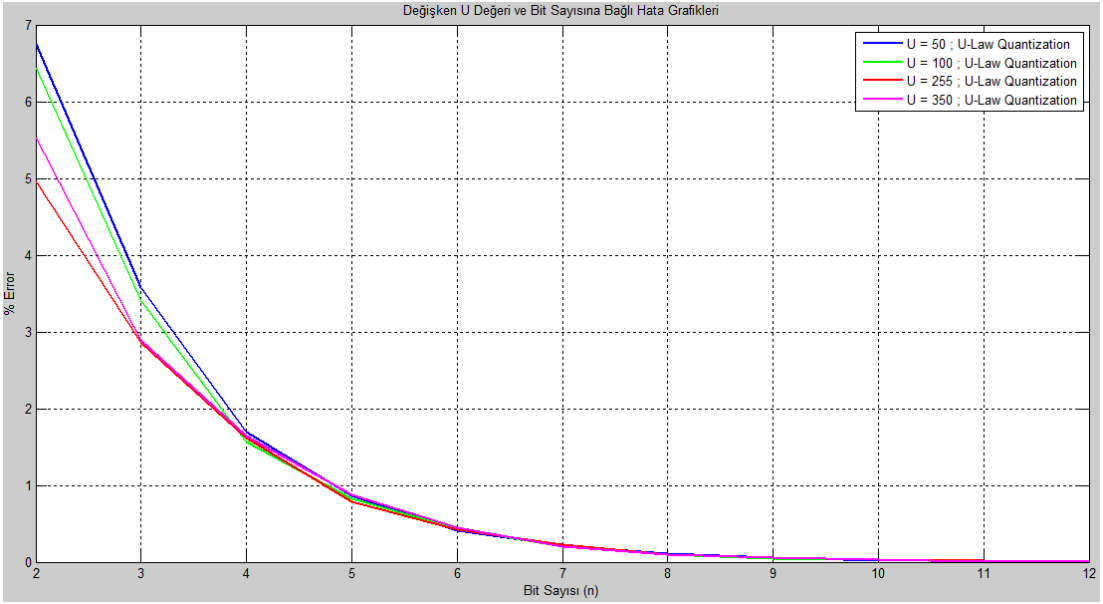
Kullanılan çeşitli sıkıştırma yöntemlerinde elde edilen optimum değerler sistemlerin minimum hata yaptığı değerlerdir. Buna ek olarak değişken bit sayısı, A ve μ değeri kullanıldığında, A-kuralı ve μ-kuralı için sıkıştırma yöntemlerinin yaptığı hatalar görülmektedir.

Şekil 4.28 ve Şekil 4.29'da μ-kuralı için değişken bit ve μ değeri için yapılan hatalar gösterilmektedir.

Her iki grafik incelendiğinde düşük bit sayıları kullanıldığında yüksek μ değerlerinde sistemin yaptığı hataların daha az olduğu görülmektedir. Yüksek bit sayılarında ise hata değerleri çok küçüldüğü için μ değerlerinin farkı çok anlaşılamamaktadır. Ancak bit sayısı artmaya başladığında optimum seviye olan 255, tipik kullanım değeri olan 8 bit seviyesinde ve 8 bite kadar olan bölgelerde en az hata yapan değerin optimum değer yani 255 olduğu görülmektedir.

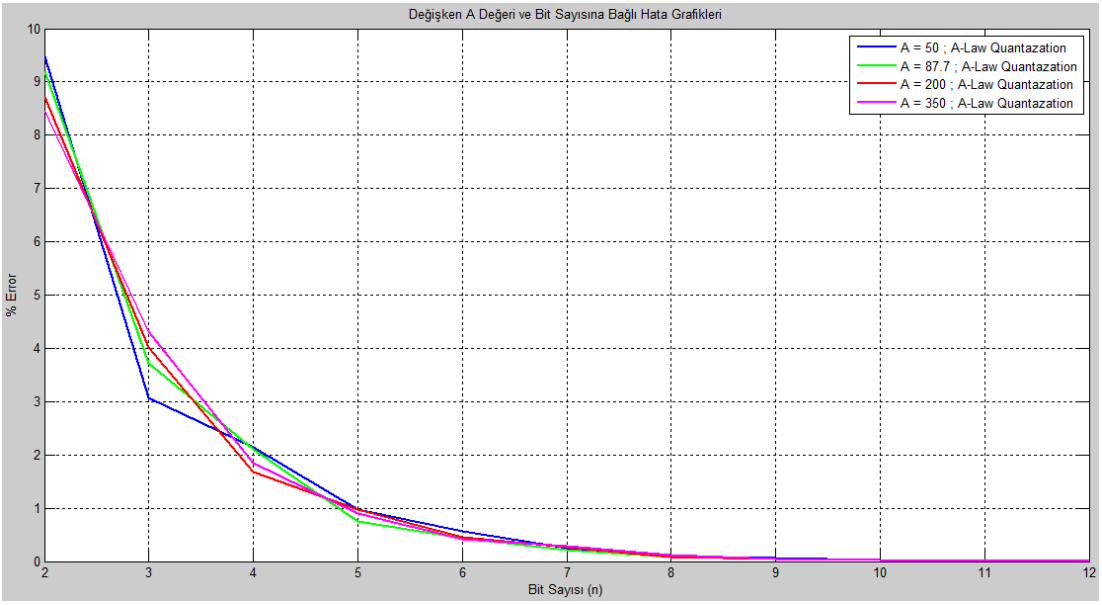


Şekil 4.28 Sinüs sinyali için değişken bit sayısı ve μ değerine bağlı hata grafiği

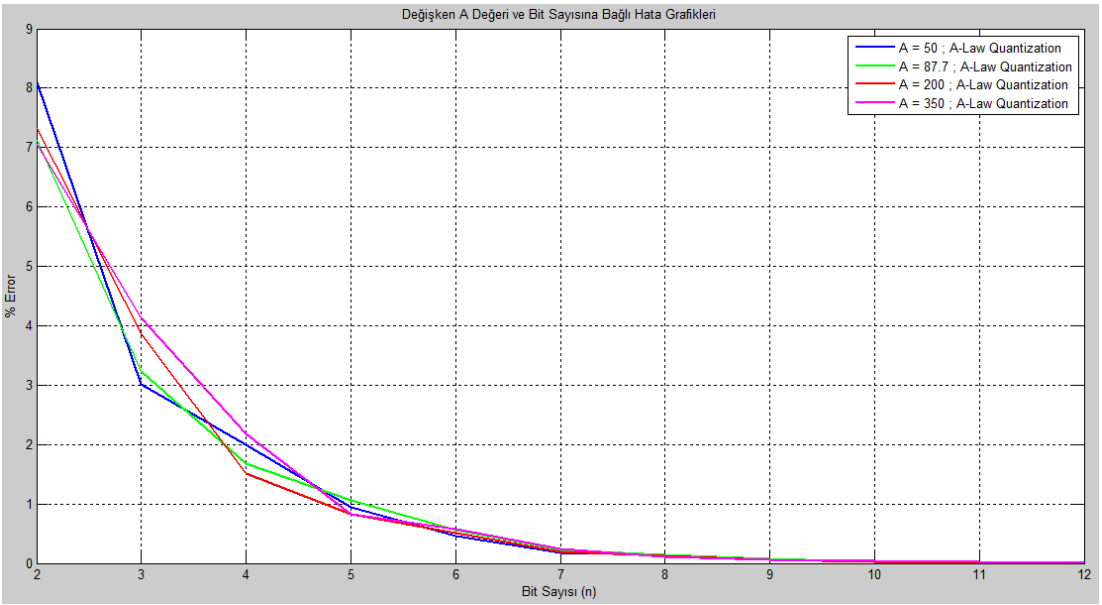


Şekil 4.29 Ses sinyali için değişken μ değerine bağlı hata grafikleri

A-kuralı için değişken bit ve A değeri kullanıldığında hesaplanan hatalar Şekil 4.30 ve Şekil 4.31'de gösterilmektedir.



Şekil 4.30 Sinüs sinyali için değişken bit sayısı ve A değerine bağlı hata grafiği



Şekil 4.31 Ses sinyali için değişken bit sayısı ve A değerine bağlı hata grafiği

Grafikler incelendiğinde 2 bit seviyesinde yüksek A değerlerinin ortalama hataları azalmaktadır. 3 bit seviyesine ulaşıldığında ise sistem tam tersine dönerek en düşük hatanın düşük A değerlerinde yapıldığı görülmektedir. Ancak 5 bit ve daha yüksek seviyelerde optimum A değeri olan 87.7 kullanılarak yapılan sıkıştırma sonucunda en düşük hata değerlerinin elde edildiği görülmektedir. Aynı bit

değerlerinde A = 50 değeri ise en yüksek hata değerine sahip olduğu görülmektedir. Değişken A değeri kullanılan grafiklerden elde edilen veriler sonucunda giriş sinyalinin önemsiz olduğu görülmektedir ve A-kuralı sıkıştırma yöntemi için optimum değerin her bit seviyesinde minimum hata elde edemediği saptanmaktadır.

5. SONUÇ

Yapılan araştırmalar ve simülasyonlar sonucunda haberleşme sistemleri ve kullanılan teknikler ile ilgili önemli veriler toplanmıştır. Teorik incelemelerde, haberleşme sistemlerinin günlük hayatımızdaki önemleri ve gereklilikleri saptanmıştır. Simülasyon çalışmalarında ise teorik olan denklem, formül, yöntem gibi kavramların pratik hayata uygun analizleri yapılmıştır.

Simülasyon çalışmalarında, A-kuralı ve μ-kuralı kullanılarak sıkıştırılan sinyallerin, sıkıştırma işlemi uygulanmadan yapılan kunatalama işlemlerine göre ortalama hatalarının azaldığı tespit edilmiştir. Sıkıştırma yöntemleri arasında yapılan simülasyonlarda ise A-kuralının daha başarılı olduğu görülmektedir. μ-kuralının ise ses sinyaline benzer formlardaki sinyallerde; yani düşük genliklerin çoğunlukta, yüksek genliklerin ise az görüldüğü sinyallerde optimum seviyenin, μ = 255 değerinin en iyi sonuç elde ettiği görülmektedir. Daha düzgün formlu sinyallerde ise yine optimum değer civarında ortalama hatanın azaldığı görülmektedir.

Kuantalama yapılırken kullanılan doğrusal kuantalamaların, doğrusal olmayan kuantalamalardan daha başarısız olduğu görülmektedir. Sinyaller örneklenirken, örnek sayısının arttırılması yapılan ortalama hataları azaltmaktadır. Bit sayılarına ilişkin yapılan analizler sonucunda ise kuantalama işlemleri yapılırken kullanılan bit sayılarının arttırılması ile ortalama hatanın azaldığı tespit edilmektedir.

Simülasyonlar sonucunda elde edilen sonuçlar:

i. Bit sayısı arttıkça kuantalama hatalarında azalma.
ii. Sıkıştırılan sinyallerin ortalama kuanta hatalarında azalma.
iii. μ-kuralının ses sinyaline benzer sinyaller için μ = 255 değerinde minimum hata değerleri elde etmesi.
iv. Kullanılan μ ve A değerlerinin optimum seviyelerinin bit sayısına göre değişmesi.
v. Doğrusal olmayan kuantalama yöntemlerinin, doğrusal kuantalama yöntemlerine göre ortalama daha az hata oranına sahip olması.

vi. Sıkıştırılan sinyallerde kullanılan bit sayılarında azalma.

vii. Sıkıştırılan sinyallerin daha doğrusal (eğim) bir form alarak kuantalama seviyelerinin düzenli dağılımında iyileşme.

KAYNAKLAR

Sarp ERTÜRK (2005). Sayısal Haberleşme. Birsen Yayınevi. Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği, Kocaeli.

Dr. Cahit KARAKUŞ (2011). Sayısal Haberleşme Sistemleri. İstanbul Kültür Üniversitesi, İstanbul.

Musa ÇIBUK (2004). Haberleşme Sistemlerinde Kullanılan Temel Kodlama ve Sıkıştırma Teknikleri. Doktora Semineri. Fırat Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Leon W. Couch (2006). Digital and Analog Communication Systems. (7th Edition). Prentice Hall, New Jersey.

Milli Eğitim Bakanlığı. (2011). Elektrik-Elektronik Teknolojisi – Analog ve Sayısal Haberleşme.
http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/Analog%20Ve%20Say%C4%B1sal%20Haberle%C5%9Fme.pdf

Milli Eğitim Bakanlığı. (2013). Bilişim Teknolojileri - Haberleşme Teknikleri.
http://megep.meb.gov.tr/mte_program_modul/moduller_pdf/Haberle%C5%9Fme%20Teknikleri.pdf

Milli Eğitim Bakanlığı. (2012). Sayısal Modülasyon.
http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf/Say%C4%B1sal%20Mod%C3%BClasyon.pdf

Haberleşme. (2014). Vikipedia. Özgür Ansiklopedi.
http://tr.wikipedia.org/wiki/%C4%B0leti%C5%9Fim

Sayısal Haberleşme. (2009). Vikipedia. Özgür Ansiklopedi.
http://tr.wikipedia.org/wiki/Say%C4%B1sal_haberle%C5%9Fme

A-Law. (2014). Vikipedia. Özgür Ansiklopedi.
http://en.wikipedia.org/wiki/A-law_algorithm

μ-Law. (2014). Vikipedia. Özgür Ansiklopedi.
http://en.wikipedia.org/wiki/%CE%9C-law_algorithm

EKLER

Sinüs Sinyali İçin PCM Simülasyonlarına İlişkin Matlab Kodları

```
clear all;
close all;
fm=1;                               % Frekans
A=1;                                % Amplitude
fs=24*fm;                           % Örnekleme hızı
n=12;                               % Bit sayısı
u=500;
Al=500;
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);                 %---Sinyal-----
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);              % xs Örneklenen sinyal
xs=awgn(xso,30000);                 % AWGN ekleniyor
e=1;                                % Hata değişkeni

for n=2 : n

   %---Lineer Kuantalama---
Mmax=max(xs);                   % Max amplitude bulunuyor
x1=real(xs)+Mmax;               % Örneklenen sinyal + amplitude
x1=x1/(2*Mmax);
L=(-1+2^n);                     % Bit seviyesi
x1=L*x1;
xq1=round(x1);                  % En yakın tam sayıya yuvarla
r1=xq1/L;
r1=2*Mmax*r1;
r1=r1-Mmax;                     %r Kuantalanmış sinyal
y1=[];
for i=1:length(xq1)
    d1=dec2base(xq1(i),2);
    y1=[y1 double(d1)-48];
end

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                       % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x2=L*x2;
xq2=round(x2);                          % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                             %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(xq2(i),2);
```

```
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                              % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                    % Bit seviyesi
x3=L*x3;
xq3=round(x3);                                 % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(xq3(i),2);
    y3=[y3 double(d3)-48];
end

%%% Non- Lineer Kuantalama %%%
Mmax=max(sign(xs).*(1/u).*((1+u).^abs(xs)-1));                    %
Max amplitude bulunuyor
x4=real(sign(xs).*(1/u).*((1+u).^abs(xs)-1))+Mmax;                %
Örneklenen sinyal + amplitude
x4=x4/(2*Mmax);
L=(-1+2^n);                                                       %
Bit seviyesi
x4=L*x4;
xq4=round(x4);                                                    %
En yakın tam sayıya yuvarla
r4=xq4/L;
r4=2*Mmax*r4;
r4=r4-Mmax;                                                       %r
Kuantalanmış sinyal
y4=[];
for i=1:length(xq4)
    d4=dec2base(xq4(i),2);
    y4=[y4 double(d4)-48];
end

%%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e1(i)=abs(x1(i)-xq1(i));
end
er1=((sum(e1)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er2=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
```

```
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er3=((sum(e3)/(fs+1))*100)/(2^n);
%--------------------------------
for i=1:(fs+1)
    e4(i)=abs(x4(i)-xq4(i));
end
er4=((sum(e4)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn1(e)=er1;
errn2(e)=er2;
errn3(e)=er3;
errn4(e)=er4;

%--------- Quantazation Level Errors ---------

LX1(2^n)=errn1(e);
LX2(2^n)=errn2(e);
LX3(2^n)=errn3(e);
LX4(2^n)=errn4(e);

e=e+1;

end

%%%%%%%%%%%%%%%%%%%


% -------- Genel Grafikler - Giriş Sinyali ve Örneklenmiş Sinyaller
-------

figure(1)
subplot(1,4,1)
plot(t,x,'linewidth',2)
title('Quantazation')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r1,'r','linewidth',2)
hold off
legend('Original Signal','Lineer Quantazation + AWGN');
grid

subplot(1,4,2)
plot(t,x,'linewidth',2)
title('Quantazation')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r2,'r','linewidth',2)
hold off
legend('Original Signal','U-Law Quantazation + AWGN');
grid

subplot(1,4,3)
plot(t,x,'linewidth',2)
```

```
title('Quantazation')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r3,'r','linewidth',2)
hold off
legend('Original Signal','A-Law Quantazation + AWGN');
grid

subplot(1,4,4)
plot(t,x,'linewidth',2)
title('Quantazation')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r4,'r','linewidth',2)
hold off
legend('Original Signal','Non-Lineer Quantazation + AWGN');
grid

figure(111)
plot(t,x,'b','linewidth',4)
hold on
plot(ts,r1,'r','linewidth',2)
hold on
plot(ts,r2,'g','linewidth',2)
hold on
plot(ts,r3,'m','linewidth',2)
hold on
plot(ts,r4,'y','linewidth',2)
title('Quantazation')
ylabel('Amplitude')
xlabel('Time (sec)')
legend('Original Signal + AWGN','Lineer Quantazation + AWGN','U-Law
Quantazation + AWGN','A-Law Quantazation + AWGN','Non-Lineer
Quantazation + AWGN');
hold off
grid

% -------- Bit Grafikleri -----------

figure(2)
subplot(4,1,1)
stairs([y1 y1(length(y1))],'linewidth',1.5)
title('Lineer Coding')
ylabel('Binary Signal')
axis([0 length(y1) -0.1 1.1])
grid

subplot(4,1,2)
stairs([y2 y2(length(y2))],'linewidth',1.5)
title('U-Law Coding')
ylabel('Binary Signal')
axis([0 length(y2) -0.1 1.1])
grid

subplot(4,1,3)
stairs([y3 y3(length(y3))],'linewidth',1.5)
title('A-Law Coding')
ylabel('Binary Signal')
```

```
axis([0 length(y3) -0.1 1.1])
grid

subplot(4,1,4)
stairs([y4 y4(length(y4))],'linewidth',1.5)
title('Non-Lineer Coding')
ylabel('Binary Signal')
xlabel('Bits')
axis([0 length(y4) -0.1 1.1])
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ----- Giriş Sinyali + AWGN -------

figure(3)
plot(ts,xs,'r','linewidth',2)
legend('Orginal Signal + AWGN');
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ---------- Anlık Kuantalama Hata Grafikleri ---------

figure(4)
subplot(4,1,1)
plot(ts,(e1*100)/(2^n),'linewidth',2)
title('Lineer Quantazation Error')
ylabel('% Error')
hold on
plot(0:0.001:1,er1,'r','linewidth',2)
legend('Lineer Quantazation Error','% Avg Error');
grid

subplot(4,1,2)
plot(ts,(e2*100)/(2^n),'linewidth',2)
title('U-Law Quantazation Error')
ylabel('% Error')
hold on
plot(0:0.001:1,er2,'r','linewidth',2)
legend('U-Law Quantazation Error','% Avg Error');
grid

subplot(4,1,3)
plot(ts,(e3*100)/(2^n),'linewidth',2)
title('A-Law Quantazation Error')
ylabel('% Error')
hold on
plot(0:0.001:1,er3,'r','linewidth',2)
legend('A-Law Quantazation Error','% Avg Error');
grid

subplot(4,1,4)
plot(ts,(e4*100)/(2^n),'linewidth',2)
title('Non-Lineer Quantazation Error')
ylabel('% Error')
xlabel('Time (sec)')
hold on
plot(0:0.001:1,er4,'r','linewidth',2)
```

```
legend('Non-Lineer Quantazation Error','% Avg Error');
grid


figure(41)
plot(ts,(e1*100)/(2^n),'b','linewidth',2)
hold on
plot(ts,(e2*100)/(2^n),'r','linewidth',2)
hold on
plot(ts,(e3*100)/(2^n),'g','linewidth',2)
hold on
plot(ts,(e4*100)/(2^n),'m','linewidth',2)
hold on
title('Quantazation Error')
ylabel('% Error')
xlabel('Time (sec)')
legend('Lineer Quantazation Error','U-Law Quantazation Error','A-Law
Quantazation Error','Non-Lineer Quantazation Error');
hold off
grid


figure(42)
plot(0:0.001:1,er1,'b','linewidth',2)
hold on
plot(0:0.001:1,er2,'r','linewidth',2)
hold on
plot(0:0.001:1,er3,'g','linewidth',2)
hold on
plot(0:0.001:1,er4,'m','linewidth',2)
title('Quantazation Avg Error')
ylabel('% Error')
xlabel('Time (sec)')
legend('Lineer Quantazation Error (Blue)','U-Law Quantazation Error
(Red)','A-Law Quantazation Error (Green)','Non-Lineer Quantazation
Error (Magenta)');
hold off
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%------------- Bit Sayısına Bağlı Hata Grafikleri --------------

figure(5)
subplot(1,4,1)
plot(2:n,errn1(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('Lineer Quantazation');
grid();

subplot(1,4,2)
plot(2:n,errn2(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('U-Law Quantazation');
grid();
```

```
subplot(1,4,3)
plot(2:n,errn3(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('A-Law Quantazation');
grid();

subplot(1,4,4)
plot(2:n,errn4(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('Non-Lineer Quantazation');
grid();


figure(51)
plot(2:n,errn1(1:n-1),'b','linewidth',2);
hold on
plot(2:n,errn2(1:n-1),'g','linewidth',2);
hold on
plot(2:n,errn3(1:n-1),'r','linewidth',2);
hold on
plot(2:n,errn4(1:n-1),'y','linewidth',2);
title('Bit Sayısına Bağlı Hata Grafikleri')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('Lineer Quantazation','U-Law Quantazation','A-Law
Quantazation','Non-Lineer Quantazation');
grid();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------- Kuanta Seviyelerine Bağlı Hata Grafikleri ---------

figure(6)
subplot(4,1,1)
stem(1:2^n,LX1(),'linewidth',2)
title('Lineer Quantazation Error')
ylabel('% Error')
grid

subplot(4,1,2)
stem(1:2^n,LX2(),'linewidth',2)
title('U-Law Quantazation Error')
ylabel('% Error')
grid

subplot(4,1,3)
stem(1:2^n,LX3(),'linewidth',2)
title('A-Law Quantazation Error')
ylabel('% Error')
grid

subplot(4,1,4)
stem(1:2^n,LX4(),'linewidth',2)
title('Non-Lineer Quantazation Error')
ylabel('% Error')
xlabel('Quantazation Levels')
```

```
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%------------- Mu Değerine Bağlı Hata Hesaplamaları --------------

n=6;                % Bit değerini sabit tutmak için ara düzeltme
u=500;
Al=500;

for u=2 : u

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                                   % Örneklenen sinyal +
amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                                         % Bit seviyesi
x2=L*x2;
xq2=round(x2);                                      % En yakın tam sayıya
yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                                         %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(xq2(i),2);
    y2=[y2 double(d2)-48];
end

%%% Non- Lineer Kuantalama %%%

Mmax=max(sign(xs).*(1/u).*((1+u).^abs(xs)-1));                          %
Max amplitude bulunuyor
x4=real(sign(xs).*(1/u).*((1+u).^abs(xs)-1))+Mmax;                      %
Örneklenen sinyal + amplitude
x4=x4/(2*Mmax);
L=(-1+2^n);                                                             %
Bit seviyesi
x4=L*x4;
xq4=round(x4);                                                          %
En yakın tam sayıya yuvarla
r4=xq4/L;
r4=2*Mmax*r4;
r4=r4-Mmax;                                                             %r
Kuantalanmış sinyal
y4=[];
for i=1:length(xq4)
    d4=dec2base(xq4(i),2);
    y4=[y4 double(d4)-48];
end

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er2=((sum(e2)/(fs+1))*100)/(2^n);
```

```
for i=1:(fs+1)
    e4(i)=abs(x4(i)-xq4(i));
end
er4=((sum(e4)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

erru2(e)=er2;           % U Değerine bağlı Hata Hesaplamaları

erru4(e)=er4;           % U Değerine bağlı Hata Hesaplamaları (Non-
Lineer)

e=e+1;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e=1;                    % e değişkenini sabit tutmak için ara
düzeltme

%------------- A Değerine Bağlı Hata Hesaplamaları --------------

for Al=2 : Al

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                       % Örneklenen sinyal + amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x3=L*x3;
xq3=round(x3);                          % En yakın tam sayıya yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(xq3(i),2);
    y3=[y3 double(d3)-48];
end

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er3=((sum(e3)/(fs+1))*100)/(2^n);
%--------------------------------

%------- Hata Matrixleri -------

erra3(e)=er3;            % A Değerine bağlı Hata Hesaplamaları

e=e+1;
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---- Mu Değerine Bağlı Hata Grafiği -----

figure(7)
subplot(2,1,1)
plot(2:u,erru2(1:u-1),'linewidth',2);
title('Mu Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Mu Değeri (U)')
legend('U-Law Quantazation');
grid();

subplot(2,1,2)
plot(2:u,erru4(1:u-1),'linewidth',2);
title('Non-Lineer Mu Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Mu Değeri (U)')
legend('Non-Lineer Quantazation');
grid();


figure(71)
plot(2:u,erru2(1:u-1),'linewidth',2);
hold on
plot(2:u,erru4(1:u-1),'r','linewidth',2);
title('U-Law ve Non-Lineer İçin Mu Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Mu Değeri (U)')
legend('U-Law Quantazation','Non-Lineer Quantazation');
grid();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---- A Değerine Bağlı Hata Grafiği -----

figure(8)
plot(2:Al,erra3(1:Al-1),'linewidth',2);
title('A Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('A Değeri (A)')
legend('A-Law Quantazation');
grid();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ------ A ve Mu Değerleri ortak Hata Grafiği -----

figure(9)
plot(2:u,erru2(1:u-1),'linewidth',2);
title('A (A) ve Mu (U) Değerine Bağlı Hata Grafiği')
hold on
plot(2:Al,erra3(1:Al-1),'r','linewidth',2);
ylabel('% Error')
xlabel('A (A) ve Mu (U) Değeri')
legend('U-Law Quantazation','A-Law Quantazation');
```

```
grid()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------- Değişken A,U ve Bit Sayısı Hesaplamaları --------

fs=24*fm;                          % Örnekleme hızı
n=12;                              % Bit sayısı
u=500;
Al=500;
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);                %---Sinyal-----
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);             % xs Örneklenen sinyal
xs=awgn(xso,30000);                % AWGN ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=50;
Al=50;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                          % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x2=L*x2;
xq2=round(x2);                             % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                                %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(xq2(i),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                                % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                      % Bit seviyesi
x3=L*x3;
xq3=round(x3);                                   % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(xq3(i),2);
```

```
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er21=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er31=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn21(e)=er21;
errn31(e)=er31;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fs=24*fm;                          % Örnekleme hızı
n=12;                              % Bit sayısı
u=500;
Al=500;
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);                %---Sinyal-----
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);             % xs Örneklenen sinyal
xs=awgn(xso,30000);                % AWGN ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=100;
Al=87.7;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                      % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                            % Bit seviyesi
x2=L*x2;
xq2=round(x2);                         % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                            %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
```

```
    d2=dec2base(xq2(i),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x3=L*x3;
xq3=round(x3);                             % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(xq3(i),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er22=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er32=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn22(e)=er22;
errn32(e)=er32;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fs=24*fm;                          % Örnekleme hızı
n=12;                              % Bit sayısı
u=500;
Al=500;
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);                %---Sinyal-----
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);             % xs Örneklenen sinyal
```

```
xs=awgn(xso,30000);              % AWGN ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=255;
Al=200;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                       % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x2=L*x2;
xq2=round(x2);                          % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                             %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(xq2(i),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                               % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                     % Bit seviyesi
x3=L*x3;
xq3=round(x3);                                  % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(xq3(i),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er23=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
```

```
end
er33=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn23(e)=er23;
errn33(e)=er33;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fs=24*fm;                          % Örnekleme hızı
n=12;                              % Bit sayısı
u=500;
Al=500;
t=0:1/(100*fm):1;
x=A*sin(2*pi*fm*t);                %---Sinyal-----
ts=0:1/fs:1;
xso=A*sin(2*pi*fm*ts);             % xs Örneklenen sinyal
xs=awgn(xso,30000);                % AWGN ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=350;
Al=350;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                      % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                            % Bit seviyesi
x2=L*x2;
xq2=round(x2);                         % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                            %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(xq2(i),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
```

```
x3=L*x3;
xq3=round(x3);                                  % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(xq3(i),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er24=((sum(e2)/(fs+1))*100)/(2^n);
%--------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er34=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn24(e)=er24;
errn34(e)=er34;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%----------- Değişken A,U ve Bit Sayısı Hata Grafikleri ----------

figure(10)
plot(2:n,errn21(1:n-1),'b','linewidth',2);
hold on
plot(2:n,errn22(1:n-1),'g','linewidth',2);
hold on
plot(2:n,errn23(1:n-1),'r','linewidth',2);
hold on
plot(2:n,errn24(1:n-1),'m','linewidth',2);
hold on
title('Değişken U Değeri ve Bit Sayısına Bağlı Hata Grafikleri')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('U = 50 ; U-Law Quantazation','U = 100 ; U-Law
Quantazation','U = 255 ; U-Law Quantazation','U = 350 ; U-Law
Quantazation');
grid();

figure(11)
plot(2:n,errn31(1:n-1),'b','linewidth',2);
hold on
```

```
plot(2:n,errn32(1:n-1),'g','linewidth',2);
hold on
plot(2:n,errn33(1:n-1),'r','linewidth',2);
hold on
plot(2:n,errn34(1:n-1),'m','linewidth',2);
hold on
title('Değişken A Değeri ve Bit Sayısına Bağlı Hata Grafikleri')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('A = 50 ; A-Law Quantazation','A = 87.7 ; A-Law
Quantazation','A = 200 ; A-Law Quantazation','A = 350 ; A-Law
Quantazation');
grid();
```

Ses Sinyali İçin PCM Simülasyonlarına İlişkin Matlab Kodları

```
clear all;
close all;
fm=1;                                % Frekans
A=1;                                 % Amplitude
fs=1000;                             % Örieklеme hızı
n=12;                                % Bit sayısı
u=500;                               % U Değeri
Al=500;                              % A Değeri
t=0:1/(1000*fm):1;
x=0.005*sin(2*pi*(fm*100)*t);        % Ses Sinyali oluşturuluyor

for i=200 : 250

    x(i)=x(i)*30;

end
for i=251 : 300

    x(i)=x(i)*50;

end

for i=400 : 450

    x(i)=x(i)*30;

end

for i=451 : 500

    x(i)=x(i)*90;

end

for i=540 : 600

    x(i)=x(i)*30;

end

for i=601 : 650

    x(i)=x(i)*80;

end
for i=651 : 700

    x(i)=x(i)*120;

end

for i=701 : 720
```

```
    x(i)=x(i)*190;

end

ts=0:1/(1000):1;
xs=0.005*sin(2*pi*(fm*100)*ts);          % Örneklenen Sinyal

for i=200 : 250

    xs(i)=xs(i)*30;

end
for i=251 : 300

    xs(i)=xs(i)*50;

end

for i=400 : 450

    xs(i)=xs(i)*30;

end
for i=451 : 500

    xs(i)=xs(i)*90;

end

for i=540 : 600

    xs(i)=xs(i)*30;

end

for i=601 : 650

    xs(i)=xs(i)*80;

end
for i=651 : 700

    xs(i)=xs(i)*120;

end

for i=701 : 720

    xs(i)=xs(i)*190;

end

xs=awgn(xs,300);                  % AWGN Ekleniyor
e=1;                              % Hata değişkeni
```

```
for n=2 : n

   %---Lineer Kuantalama---

Mmax=max(xs);                % Max amplitude bulunuyor
x1=real(xs)+Mmax;            % Örneklenen sinyal + amplitude
x1=x1/(2*Mmax);
L=(-1+2^n);                  % Bit seviyesi
x1=L*x1;
xq1=round(x1);               % En yakın tam sayıya yuvarla
r1=xq1/L;
r1=2*Mmax*r1;
r1=r1-Mmax;                  %r Kuantalanmış sinyal
y1=[];
for i=1:length(xq1)
    d1=dec2base(abs(xq1(i)),2);
    y1=[y1 double(d1)-48];
end

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                    % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                          % Bit seviyesi
x2=L*x2;
xq2=round(x2);                       % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                          %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(abs(xq2(i)),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x3=L*x3;
xq3=round(x3);                             % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(abs(xq3(i)),2);
    y3=[y3 double(d3)-48];
end

%%% Non- Lineer Kuantalama %%%
```

```
Mmax=max(sign(xs).*(1/u).*((1+u).^abs(xs)-1));                    %
Max amplitude bulunuyor
x4=real(sign(xs).*(1/u).*((1+u).^abs(xs)-1))+Mmax;                %
Örneklenen sinyal + amplitude
x4=x4/(2*Mmax);
L=(-1+2^n);                                                       %
Bit seviyesi
x4=L*x4;
xq4=round(x4);                                                    %
En yakın tam sayıya yuvarla
r4=xq4/L;
r4=2*Mmax*r4;
r4=r4-Mmax;                                                       %r
Kuantalanmış sinyal
y4=[];
for i=1:length(xq4)
    d4=dec2base(abs(xq4(i)),2);
    y4=[y4 double(d4)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e1(i)=abs(x1(i)-xq1(i));
end
er1=((sum(e1)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er2=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er3=((sum(e3)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e4(i)=abs(x4(i)-xq4(i));
end
er4=((sum(e4)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn1(e)=er1;
errn2(e)=er2;
errn3(e)=er3;
errn4(e)=er4;

%--------- Quantization Level Errors ---------

LX1(2^n)=errn1(e);
LX2(2^n)=errn2(e);
LX3(2^n)=errn3(e);
LX4(2^n)=errn4(e);

e=e+1;
```

```
end

%%%%%%%%%%%%%%%%%%%

% -------- Genel Grafikler - Giriş Sinyali ve Örneklenmiş Sinyaller
-------

figure(1)
subplot(1,4,1)
plot(t,x,'linewidth',2)
title('Quantization')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r1,'r','linewidth',2)
hold off
legend('Original Signal','Lineer Quantization + AWGN');
grid

subplot(1,4,2)
plot(t,x,'linewidth',2)
title('Quantization')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r2,'r','linewidth',2)
hold off
legend('Original Signal','U-Law Quantization + AWGN');
grid

subplot(1,4,3)
plot(t,x,'linewidth',2)
title('Quantization')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r3,'r','linewidth',2)
hold off
legend('Original Signal','A-Law Quantization + AWGN');
grid

subplot(1,4,4)
plot(t,x,'linewidth',2)
title('Quantization')
ylabel('Amplitude')
xlabel('Time (sec)')
hold on
plot(ts,r4,'r','linewidth',2)
hold off
legend('Original Signal','Non-Lineer Quantization + AWGN');
grid

figure(111)
plot(t,x,'b','linewidth',4)
hold on
plot(ts,r1,'r','linewidth',2)
hold on
plot(ts,r2,'g','linewidth',2)
hold on
```

```
plot(ts,r3,'m','linewidth',2)
hold on
plot(ts,r4,'y','linewidth',2)
title('Quantization')
ylabel('Amplitude')
xlabel('Time (sec)')
legend('Original Signal + AWGN','Lineer Quantization + AWGN','U-Law
Quantization + AWGN','A-Law Quantization + AWGN','Non-Lineer
Quantization + AWGN');
hold off
grid

% -------- Bit Grafikleri -----------

figure(2)
subplot(4,1,1)
stairs([y1 y1(length(y1))],'linewidth',1.5)
title('Lineer Coding')
ylabel('Binary Signal')
axis([0 length(y1) -0.1 1.1])
grid

subplot(4,1,2)
stairs([y2 y2(length(y2))],'linewidth',1.5)
title('U-Law Coding')
ylabel('Binary Signal')
axis([0 length(y2) -0.1 1.1])
grid

subplot(4,1,3)
stairs([y3 y3(length(y3))],'linewidth',1.5)
title('A-Law Coding')
ylabel('Binary Signal')
axis([0 length(y3) -0.1 1.1])
grid

subplot(4,1,4)
stairs([y4 y4(length(y4))],'linewidth',1.5)
title('Non-Lineer Coding')
ylabel('Binary Signal')
xlabel('Bits')
axis([0 length(y4) -0.1 1.1])
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ----- Giriş Sinyali + AWGN -------

figure(3)
plot(ts,xs,'r','linewidth',2)
legend('Orginal Signal + AWGN');
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ---------- Anlık Kuantalama Hata Grafikleri ---------

figure(4)
subplot(4,1,1)
```

```
plot(ts,(e1*100)/(2^n),'linewidth',2)
title('Lineer Quantization Error')
ylabel('% Error')
hold on
plot(0:0.001:1,er1,'r','linewidth',2)
legend('Lineer Quantization Error','% Avg Error');
grid

subplot(4,1,2)
plot(ts,(e2*100)/(2^n),'linewidth',2)
title('U-Law Quantization Error')
ylabel('% Error')
hold on
plot(0:0.001:1,er2,'r','linewidth',2)
legend('U-Law Quantization Error','% Avg Error');
grid

subplot(4,1,3)
plot(ts,(e3*100)/(2^n),'linewidth',2)
title('A-Law Quantization Error')
ylabel('% Error')
hold on
plot(0:0.001:1,er3,'r','linewidth',2)
legend('A-Law Quantization Error','% Avg Error');
grid

subplot(4,1,4)
plot(ts,(e4*100)/(2^n),'linewidth',2)
title('Non-Lineer Quantization Error')
ylabel('% Error')
xlabel('Time (sec)')
hold on
plot(0:0.001:1,er4,'r','linewidth',2)
legend('Non-Lineer Quantization Error','% Avg Error');
grid


figure(41)
plot(ts,(e1*100)/(2^n),'b','linewidth',2)
hold on
plot(ts,(e2*100)/(2^n),'r','linewidth',2)
hold on
plot(ts,(e3*100)/(2^n),'g','linewidth',2)
hold on
plot(ts,(e4*100)/(2^n),'m','linewidth',2)
hold on
title('Quantization Error')
ylabel('% Error')
xlabel('Time (sec)')
legend('Lineer Quantization Error','U-Law Quantization Error','A-Law
Quantization Error','Non-Lineer Quantization Error');
hold off
grid


figure(42)
stem(1,er1,'b','linewidth',3)
axis([0,5,0,0.01])
hold on
stem(2,er2,'r','linewidth',3)
```

```
axis([0,5,0,0.01])
hold on
stem(3,er3,'g','linewidth',3)
axis([0,5,0,0.01])
hold on
stem(4,er4,'m','linewidth',3)
axis([0,5,0,0.01])
title('Quantization Avg Error')
ylabel('% Error')
xlabel('Order')
legend('1. Lineer Quantization Avg Error','2. U-Law Quantization Avg
Error','3. A-Law Quantization Avg Error','4. Non-Lineer Quantization
Avg Error');
hold off
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%------------- Bit Sayısına Bağlı Hata Grafikleri --------------

figure(5)
subplot(1,4,1)
plot(2:n,errn1(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('Lineer Quantization');
grid();

subplot(1,4,2)
plot(2:n,errn2(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('U-Law Quantization');
grid();

subplot(1,4,3)
plot(2:n,errn3(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('A-Law Quantization');
grid();

subplot(1,4,4)
plot(2:n,errn4(1:n-1),'linewidth',2);
title('Bit Sayısına Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('Non-Lineer Quantization');
grid();


figure(51)
plot(2:n,errn1(1:n-1),'b','linewidth',2);
hold on
plot(2:n,errn2(1:n-1),'g','linewidth',2);
hold on
plot(2:n,errn3(1:n-1),'r','linewidth',2);
```

```
hold on
plot(2:n,errn4(1:n-1),'y','linewidth',2);
title('Bit Sayısına Bağlı Hata Grafikleri')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('Lineer Quantization','U-Law Quantization','A-Law
Quantization','Non-Lineer Quantization');
grid();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------- Kuanta Seviyelerine Bağlı Hata Grafikleri ---------

figure(6)
subplot(4,1,1)
stem(1:2^n,LX1(),'linewidth',2)
title('Lineer Quantization Error')
ylabel('% Error')
grid

subplot(4,1,2)
stem(1:2^n,LX2(),'linewidth',2)
title('U-Law Quantization Error')
ylabel('% Error')
grid

subplot(4,1,3)
stem(1:2^n,LX3(),'linewidth',2)
title('A-Law Quantization Error')
ylabel('% Error')
grid

subplot(4,1,4)
stem(1:2^n,LX4(),'linewidth',2)
title('Non-Lineer Quantization Error')
ylabel('% Error')
xlabel('Quantization Levels')
grid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%------------- Mu Değerine Bağlı Hata Hesaplamaları --------------

n=6;                % Bit değerini sabit tutmak için ara düzeltme
u=500;
Al=500;

for u=2 : u

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                                % Örneklenen sinyal +
amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                                      % Bit seviyesi
x2=L*x2;
xq2=round(x2);                                   % En yakın tam sayıya
yuvarla
```

```
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                                    %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(abs(xq2(i)),2);
    y2=[y2 double(d2)-48];
end

%%% Non- Lineer Kuantalama %%%

Mmax=max(sign(xs).*(1/u).*((1+u).^abs(xs)-1));                    %
Max amplitude bulunuyor
x4=real(sign(xs).*(1/u).*((1+u).^abs(xs)-1))+Mmax;                %
Örneklenen sinyal + amplitude
x4=x4/(2*Mmax);
L=(-1+2^n);                                                       %
Bit seviyesi
x4=L*x4;
xq4=round(x4);                                                    %
En yakın tam sayıya yuvarla
r4=xq4/L;
r4=2*Mmax*r4;
r4=r4-Mmax;                                                       %r
Kuantalanmış sinyal
y4=[];
for i=1:length(xq4)
    d4=dec2base(abs(xq4(i)),2);
    y4=[y4 double(d4)-48];
end

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er2=((sum(e2)/(fs+1))*100)/(2^n);

for i=1:(fs+1)
    e4(i)=abs(x4(i)-xq4(i));
end
er4=((sum(e4)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

erru2(e)=er2;           % U Değerine bağlı Hata Hesaplamaları

erru4(e)=er4;           % U Değerine bağlı Hata Hesaplamaları (Non-
Lineer)

e=e+1;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e=1;                    % e değişkenini sabit tutmak için ara
düzeltme

%------------- A Değerine Bağlı Hata Hesaplamaları --------------
```

```
for Al=2 : Al

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                       % Örneklenen sinyal + amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x3=L*x3;
xq3=round(x3);                          % En yakın tam sayıya yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(abs(xq3(i)),2);
    y3=[y3 double(d3)-48];
end

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er3=((sum(e3)/(fs+1))*100)/(2^n);
%--------------------------------

%------- Hata Matrixleri -------

erra3(e)=er3;             % A Değerine bağlı Hata Hesaplamaları

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---- Mu Değerine Bağlı Hata Grafiği -----

figure(7)
subplot(2,1,1)
plot(2:u,erru2(1:u-1),'linewidth',2);
title('Mu Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Mu Değeri (U)')
legend('U-Law Quantization');
grid();

subplot(2,1,2)
plot(2:u,erru4(1:u-1),'linewidth',2);
title('Non-Lineer Mu Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Mu Değeri (U)')
legend('Non-Lineer Quantization');
grid();
```

```
figure(71)
plot(2:u,erru2(1:u-1),'linewidth',2);
hold on
plot(2:u,erru4(1:u-1),'r','linewidth',2);
title('U-Law ve Non-Lineer İçin Mu Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('Mu Değeri (U)')
legend('U-Law Quantization','Non-Lineer Quantization');
grid();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---- A Değerine Bağlı Hata Grafiği -----

figure(8)
plot(2:Al,erra3(1:Al-1),'linewidth',2);
title('A Değerine Bağlı Hata Grafiği')
ylabel('% Error')
xlabel('A Değeri (A)')
legend('A-Law Quantization');
grid();

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ------ A ve Mu Değerleri ortak Hata Grafiği -----

figure(9)
plot(2:u,erru2(1:u-1),'linewidth',2);
title('A (A) ve Mu (U) Değerine Bağlı Hata Grafiği')
hold on
plot(2:Al,erra3(1:Al-1),'r','linewidth',2);
ylabel('% Error')
xlabel('A (A) ve Mu (U) Değeri')
legend('U-Law Quantization','A-Law Quantization');
grid()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%----------- Değişken A,U ve Bit Sayısı Hesaplamaları --------

fs=1000;                          % Örieklеme hızı
n=12;                             % Bit sayısı
u=500;                            % U Değeri
Al=500;                           % A Değeri
t=0:1/(1000*fm):1;
x=0.005*sin(2*pi*(fm*100)*t);     % Ses Sinyali oluşturuluyor

for i=200 : 250

    x(i)=x(i)*30;

end
for i=251 : 300

    x(i)=x(i)*50;
```

```
end

for i=400 : 450

    x(i)=x(i)*30;

end

for i=451 : 500

    x(i)=x(i)*90;

end

for i=540 : 600

    x(i)=x(i)*30;

end

for i=601 : 650

    x(i)=x(i)*80;

end
for i=651 : 700

    x(i)=x(i)*120;

end

for i=701 : 720

    x(i)=x(i)*190;

end

ts=0:1/(1000):1;
xs=0.005*sin(2*pi*(fm*100)*ts);          % Örneklenen Sinyal

for i=200 : 250

    xs(i)=xs(i)*30;

end
for i=251 : 300

    xs(i)=xs(i)*50;

end

for i=400 : 450

    xs(i)=xs(i)*30;

end
```

```
for i=451 : 500

    xs(i)=xs(i)*90;

end

for i=540 : 600

    xs(i)=xs(i)*30;

end

for i=601 : 650

    xs(i)=xs(i)*80;

end
for i=651 : 700

    xs(i)=xs(i)*120;

end

for i=701 : 720

    xs(i)=xs(i)*190;

end

xs=awgn(xs,300);                % AWGN Ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=50;
Al=50;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                   % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                         % Bit seviyesi
x2=L*x2;
xq2=round(x2);                      % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                         %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(abs(xq2(i)),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
```

```
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x3=L*x3;
xq3=round(x3);                             % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(abs(xq3(i)),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er21=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er31=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn21(e)=er21;
errn31(e)=er31;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fs=1000;                          % Öriekleme hızı
n=12;                             % Bit sayısı
u=500;                            % U Değeri
Al=500;                           % A Değeri
t=0:1/(1000*fm):1;
x=0.005*sin(2*pi*(fm*100)*t);     % Ses Sinyali oluşturuluyor

for i=200 : 250

    x(i)=x(i)*30;

end
for i=251 : 300
```

```
    x(i)=x(i)*50;

end

for i=400 : 450

    x(i)=x(i)*30;

end

for i=451 : 500

    x(i)=x(i)*90;

end

for i=540 : 600

    x(i)=x(i)*30;

end

for i=601 : 650

    x(i)=x(i)*80;

end
for i=651 : 700

    x(i)=x(i)*120;

end

for i=701 : 720

    x(i)=x(i)*190;

end

ts=0:1/(1000):1;
xs=0.005*sin(2*pi*(fm*100)*ts);          % Örneklenen Sinyal

for i=200 : 250

    xs(i)=xs(i)*30;

end
for i=251 : 300

    xs(i)=xs(i)*50;

end

for i=400 : 450

    xs(i)=xs(i)*30;
```

```
end
for i=451 : 500

    xs(i)=xs(i)*90;

end

for i=540 : 600

    xs(i)=xs(i)*30;

end

for i=601 : 650

    xs(i)=xs(i)*80;

end
for i=651 : 700

    xs(i)=xs(i)*120;

end

for i=701 : 720

    xs(i)=xs(i)*190;

end

xs=awgn(xs,300);                % AWGN Ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=100;
Al=87.7;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                       % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x2=L*x2;
xq2=round(x2);                          % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                             %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(abs(xq2(i)),2);
    y2=[y2 double(d2)-48];
end
```

```
% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x3=L*x3;
xq3=round(x3);                             % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(abs(xq3(i)),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er22=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er32=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn22(e)=er22;
errn32(e)=er32;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fs=1000;                           % Öriekleme hızı
n=12;                              % Bit sayısı
u=500;                             % U Değeri
Al=500;                            % A Değeri
t=0:1/(1000*fm):1;
x=0.005*sin(2*pi*(fm*100)*t);      % Ses Sinyali oluşturuluyor

for i=200 : 250

    x(i)=x(i)*30;
```

```
end
for i=251 : 300

    x(i)=x(i)*50;

end

for i=400 : 450

    x(i)=x(i)*30;

end

for i=451 : 500

    x(i)=x(i)*90;

end

for i=540 : 600

    x(i)=x(i)*30;

end

for i=601 : 650

    x(i)=x(i)*80;

end
for i=651 : 700

    x(i)=x(i)*120;

end

for i=701 : 720

    x(i)=x(i)*190;

end

ts=0:1/(1000):1;
xs=0.005*sin(2*pi*(fm*100)*ts);          % Örneklenen Sinyal

for i=200 : 250

    xs(i)=xs(i)*30;

end
for i=251 : 300

    xs(i)=xs(i)*50;

end
```

```
for i=400 : 450

    xs(i)=xs(i)*30;

end
for i=451 : 500

    xs(i)=xs(i)*90;

end

for i=540 : 600

    xs(i)=xs(i)*30;

end

for i=601 : 650

    xs(i)=xs(i)*80;

end
for i=651 : 700

    xs(i)=xs(i)*120;

end

for i=701 : 720

    xs(i)=xs(i)*190;

end

xs=awgn(xs,300);                % AWGN Ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=255;
Al=200;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                       % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x2=L*x2;
xq2=round(x2);                          % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
r2=r2-Mmax;                             %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
```

```
    d2=dec2base(abs(xq2(i)),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x3=L*x3;
xq3=round(x3);                             % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(abs(xq3(i)),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er23=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er33=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn23(e)=er23;
errn33(e)=er33;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fs=1000;                          % Öriekleme hızı
n=12;                             % Bit sayısı
u=500;                            % U Değeri
Al=500;                           % A Değeri
t=0:1/(1000*fm):1;
x=0.005*sin(2*pi*(fm*100)*t);     % Ses Sinyali oluşturuluyor

for i=200 : 250
```

```
    x(i)=x(i)*30;

end
for i=251 : 300

    x(i)=x(i)*50;

end

for i=400 : 450

    x(i)=x(i)*30;

end

for i=451 : 500

    x(i)=x(i)*90;

end

for i=540 : 600

    x(i)=x(i)*30;

end

for i=601 : 650

    x(i)=x(i)*80;

end
for i=651 : 700

    x(i)=x(i)*120;

end

for i=701 : 720

    x(i)=x(i)*190;

end

ts=0:1/(1000):1;
xs=0.005*sin(2*pi*(fm*100)*ts);          % Örneklenen Sinyal

for i=200 : 250

    xs(i)=xs(i)*30;

end
for i=251 : 300

    xs(i)=xs(i)*50;
```

```
end

for i=400 : 450

    xs(i)=xs(i)*30;

end
for i=451 : 500

    xs(i)=xs(i)*90;

end

for i=540 : 600

    xs(i)=xs(i)*30;

end

for i=601 : 650

    xs(i)=xs(i)*80;

end
for i=651 : 700

    xs(i)=xs(i)*120;

end

for i=701 : 720

    xs(i)=xs(i)*190;

end

xs=awgn(xs,300);                % AWGN Ekleniyor
e=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for n=2 : n

u=350;
Al=350;

% ---U-Law Kuantalama ---
Mn=xs/Mmax;
ux=sign(xs).*log(1+u.*Mn)./(log(1+u));
x2=real(ux)+Mmax;                       % Örneklenen sinyal + amplitude
x2=x2/(2*Mmax);
L=(-1+2^n);                             % Bit seviyesi
x2=L*x2;
xq2=round(x2);                          % En yakın tam sayıya yuvarla
r2=xq2/L;
r2=2*Mmax*r2;
```

```
r2=r2-Mmax;                                %r Kuantalanmış sinyal
y2=[];
for i=1:length(xq2)
    d2=dec2base(abs(xq2(i)),2);
    y2=[y2 double(d2)-48];
end

% ---A-Law Kuantalama ---
if (0 <= Mn < (1/Al))
ax=Al.*Mn/(1+log(Al));
elseif ((1/Al) <= Mn <= 1)
ax=sign(xs).*log(1+Al.*Mn)/(1+log(Al));
end
x3=real(ax)+Mmax;                          % Örneklenen sinyal +
amplitude
x3=x3/(2*Mmax);
L=(-1+2^n);                                % Bit seviyesi
x3=L*x3;
xq3=round(x3);                             % En yakın tam sayıya
yuvarla
r3=xq3/L;
r3=2*Mmax*r3;
r3=r3-Mmax;
y3=[];
for i=1:length(xq3)
    d3=dec2base(abs(xq3(i)),2);
    y3=[y3 double(d3)-48];
end

%%%%%%%%%%%%%%%%%%%%

% ---- Hata Hesaplamaları ----

for i=1:(fs+1)
    e2(i)=abs(x2(i)-xq2(i));
end
er24=((sum(e2)/(fs+1))*100)/(2^n);
%---------------------------------
for i=1:(fs+1)
    e3(i)=abs(x3(i)-xq3(i));
end
er34=((sum(e3)/(fs+1))*100)/(2^n);

%------- Hata Matrixleri -------

errn24(e)=er24;
errn34(e)=er34;

e=e+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%----------- Değişken A,U ve Bit Sayısı Hata Grafikleri ----------

figure(10)
plot(2:n,errn21(1:n-1),'b','linewidth',2);
hold on
```

```
plot(2:n,errn22(1:n-1),'g','linewidth',2);
hold on
plot(2:n,errn23(1:n-1),'r','linewidth',2);
hold on
plot(2:n,errn24(1:n-1),'m','linewidth',2);
hold on
title('Değişken U Değeri ve Bit Sayısına Bağlı Hata Grafikleri')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('U = 50 ; U-Law Quantization','U = 100 ; U-Law
Quantization','U = 255 ; U-Law Quantization','U = 350 ; U-Law
Quantization');
grid();

figure(11)
plot(2:n,errn31(1:n-1),'b','linewidth',2);
hold on
plot(2:n,errn32(1:n-1),'g','linewidth',2);
hold on
plot(2:n,errn33(1:n-1),'r','linewidth',2);
hold on
plot(2:n,errn34(1:n-1),'m','linewidth',2);
hold on
title('Değişken A Değeri ve Bit Sayısına Bağlı Hata Grafikleri')
ylabel('% Error')
xlabel('Bit Sayısı (n)')
legend('A = 50 ; A-Law Quantization','A = 87.7 ; A-Law
Quantization','A = 200 ; A-Law Quantization','A = 350 ; A-Law
Quantization');
grid();
```

ÖZGEÇMİŞ

Kişisel Bilgiler:

Adı, Soyadı : Hakan KAHRAMAN

Doğum Yeri : Tekirdağ / Muratlı

Doğum Tarihi : 20.07.1988

Medeni Hali : Bekâr

Yabancı Dil : İngilizce

E-Posta : hakan.kahraman.59@gmail.com

Telefon : 0 555 598 83 35

1988 yılında doğdu. İlk, Orta ve Lise Öğrenimini Tekirdağ'da tamamladı. Anadolu Lisesi Fen Bölümünden 2007 yılında mezun oldu. 2008 yılında Girne American University, Foundation Education Program (FEP) mezun oldu. 2012 yılında Girne American University, Electrical Electronics Engineering bölümünden mezun oldu. 2012 yılında Bilginoğlu Endüstri firmasında, Uygulama Mühendisi olarak başladığı görevinde çalışmaya devam etmektedir.