KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ÖRNEKLENMİŞ İŞARET İÇİN BULANIK MANTIĞA DAYALI ARADEĞERLEME ALGORİTMASI

Elk. Müh. Bekir DİZDAROĞLU

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"Elektronik Yüksek Mühendisi"
Ünvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 06. 01. 1998
Tezin Savunma Tarihi : 04. 02. 1998

Tez Danışmanı	: Yrd. Doç. Dr. Ali GANGAL
Jüri Üyesi	: Yrd. Doç. Dr. Temel KAYIKÇIOĞLU
Jüri Üyesi	: Doç. Dr. İsmail Hakkı ALTAŞ
Enstitü Müdürü	: Prof. Dr. Fazlı ARSLAN

Ocak 1998
Trabzon

ÖNSÖZ

Bu çalışmada, iletim veya depolama sırasında ayrık işaretlerin örnek değerlerinin gürültü veya başka sebeplerden dolayı bozulması durumunda, işaretin örnek değerlerinin yaklaşık olarak tekrar elde edilmesi için yeni bir bulanık mantık yöntemi geliştirilmiştir. Bulanık mantığın bu alanda pek fazla kullanılmamasından dolayı sadece yapılan bir çalışma ile karşılaştırma yapılmış ve kestirilen örnek değerlerinin her iki yöntemle performans analizi yapılmıştır.

Yüksek lisans tezi danışmanlığımı yapan ve çalışmam boyunca yardımı esirgemeyen saygıdeğer hocam Yrd. Doç. Dr. Ali GANGAL 'a teşekkürü bir borç bilirim. Ayrıca bu çalışmada yardımcı olan hocam Yrd. Doç. Dr. Temel KAYIKÇIOĞLU'na da teşekkür ederim.

Bekir DİZDAROĞLU

İÇİNDEKİLER

ÖZET

Günümüzün gelişen teknolojileri artık geleneksel elektronik denetim birimlerinden yeteri kadar verim alamamaktadır. Gün geçtikçe ortaya çıkan daha hassas birimler ve kaçınılmaz olan enerjiden tasarruf sağlama zorunluluğu bilim adamlarını bu yönde araştırmalar yapmaya itmiştir. Gitgide mükemmele yaklaşma isteği ve doğanın belki de bir gün aynısının yapay yollarla ortaya çıkarılmaya çalışılması, yapay zeka, yapay sinir ağları, çok değerli mantık ve bunlarla birlikte bulanık mantığın ortaya çıkarılmasına neden olmuştur.

Bulanık mantık her gün kullandığımız ve davranışlarımızı yorumladığımız yapıya ulaşmamızı sağlayan matematiksel bir disiplindir. Temelini doğru ve yanlış değerlerin belirlediği bulanık küme kuramı oluşturur. Burada yine geleneksel mantıkta olduğu gibi bir (1) ve sıfır (0) değeri vardır. Ancak bulanık mantık yalnızca bu değerlerle yetinmeyip, bunların ara değerlerini de kullanarak; örneğin bir uzaklığın yalnızca yakın ya da uzak olduğunu belirtmekle kalmayıp, ne kadar yakın ya da ne kadar uzak olduğunu da söyler.

Bu tezde, iletim veya depolama sırasında örnek değerleri gürültü veya başka nedenlerden dolayı bozulmuş olan bir işaretin tekrar elde edilmesinde, bulanık mantığa dayalı bir aradeğerleme algoritması geliştirilmiştir. Sonuçlar, örnek değerleri belirli bir şekilde seçilmiş olan gürültüsüz ve toplamsal gürültülü işaretler için ayrı ayrı elde edilmiş ve bozulan örnek değerlerin kestirilme performansı belirlenmiştir.

Anahtar kelimeler: Bulanık Mantık, Bulanık Küme, Elektronik Kontrol Biçimi, Yapay Zeka, Çokdeğerli Mantık, Sinir Ağları, Aradeğerleme.

SUMMARY

FUZZY LOGIC RULE - BASED INTERPOLATION ALGORITM FOR SAMPLED SIGNAL

Today's developing technologies can no more get enough benefit and output from traditional electronic control units. Increase in more sensitive and complex units in daily basis and inevitable obligation for conservation of energy compelled the scientists to do research in that direction. Increasing demand for coming close to perfection and work for may be one day to create something as identical as of nature caused to have introduced artificial intelligence, neural networks, multiple valued logic and nevertheless fuzzy logic.

The fuzzy logic is mathematical discipline that we use everyday life and provide us to reach the state of interpreting our behaviours. The fundamental of fuzzy logic is constituted of fuzzy set theory which is determined by true or false values. Here, as in traditional logic, there are one (1) and zero (0) values. However, fuzzy logic not only will satisfy with these values, but also it uses the interval values in between of those; for example, it does not only tell that the distance is far or near , but also says how near or how far it is.

In this study, in order to reconstruct the signal of which sampled values were damaged by noise or other reason during transmitting or storing, an interpolation method which is based on fuzzy logic has been developed. The results were found for each noise-free and global noisy signals which their sampled values are chosen in a certain way and the estimation performance of vanishing sampled values was determined.

Key Words: Fuzzy Logic, Fuzzy Set, Electronic Control Unit, Artificial Intelligence, Multiple Valued Logic, Neural Network, Interpolation.

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

A_i : Bulanık mantık kümesi

b : Bir sabit değer

c : Bir sabit değer

D_i : i . dikdörtgen

f : Frekans değeri

$f_i(x)$: i . doğru için fonksiyon

$f_{ö}$: Örnekleme frekansı

g_i : i . noktanın değeri

G_i : P_i'in ağırlık merkezi

k(x) : Zaman domeninde bir işaret

$\hat{k}(x)$: Bulanık mantık yöntemiyle kestirilen değer

m_i : i . noktadaki eğim

n : Alınan örnek sayısı

P_i : Bir dikdörtgensel parça

s_i : i . noktada bir eğim değeri

t_i : Yatay eksende i . nokta

v(x) : İstatistiği bilinmeyen toplamsal gürültü işareti

V : Bir koordinat

y(x) : Gürültü ile bozulmuş çıkış işareti

$\hat{y}(x)$: Aradeğerlenmiş çıkış işareti

Y : Bir koordinat

u : Üyelik fonksiyonu

(x_i, y_i) : Giriş - çıkış veri çiftleri

W : Bir koordinat

Z : Bir koordinat

$\mu_i(x)$: i . bulanık mantık üyelik fonksiyonu (işlevi)

$\sigma^2{}_i$: Orijinal işaretin değişintisi

$\sigma^2{}_{i-a}$: Aradeğerlenen işaretle orijinal işaret arasındaki hatanın değişintisi

Alt İndisler

i , j : 1, 2, 3, ... şeklinde artan değerler olarak kullanılmıştır.

1. GENEL BİLGİLER

1.1. Giriş

İşaret örneklemede, örnek değerlerinin azaltılıp iletilmesi veya depolanması oldukça önemlidir. Çünkü verilerin daha az olması, işlemin daha az zamanda ve daha hızlı bir şekilde yapılmasını sağlar. Bir de işaretin iletilmesi esnasında, işarete gürültü eklenmesinden dolayı işarette bozulmalar meydana gelmektedir. Bütün bu olumsuz durumların önlenmesi için çeşitli yöntemler geliştirilmiştir [1]. Gürültü süzgeçlemede Kalman süzgeçleme gibi süzgeçleme yöntemleri vardır [2]. Fakat bu gibi yöntemlerde hem daha fazla karmaşık işlemler yapılmasına, hem de işarete eklenen gürültünün istatistiğinin bilinmesine gerek duyulmaktadır. Bu durumda daha fazla zaman harcanmaktadır.

Bulanık mantığa dayalı yöntemler ise, bu alanda daha yeni yeni uygulanmaya başlanmıştır [3, 4]. Bulanık mantığın uygulama alanı daha çok kontrol üzerinedir. İşaret örneklemede pek nadir uygulanmıştır. Fakat geliştirilen yöntemler basit olmasına rağmen oldukça iyi sonuçlar vermektedir. Kaldı ki bulanık mantığın diğer uygulama alanlarında da oldukça iyi sonuçlar verdiği gözlenmiştir. Geliştirilen ve performans analizi için karşılaştırılması yapılan her iki yöntemde de elde edilen sonuçlar oldukça iyidir. Bulanık mantığa dayalı bu iki yöntemde de örnek değerleri parça parça işleme tabi tutulduklarından bütün örnek değerlerinin bilinmesine gerek duyulmamaktadır. Burada bir diğer önemli konu da, örneklemenin band genişliğinin iki katından (Nyquist sınırı) daha düşük bir frekansta yapılması, frekans örtüşmesinin meydana gelmesine sebep olmaktadır. Bu durum bulanık mantığa dayalı algoritmada da mevcuttur. Örnek değerleri seyrekleştirilerek yapılan işlemlerde kestirilen işarette bozulmalar meydana geldiği görülmüştür. Zaten örnekleme frekansı band genişliğinin iki katından daha az olursa, elde edilen işaretin orijinal işareti göstermesi düşünülemez.

Yapılan çalışmalarda, çıkışta elde edilecek işaretin en iyi şekilde kestirilmesi amaçlanmıştır. Bulanık mantığa dayalı yapılan çalışmalar oldukça azdır. Bu çalışmalarda yapılanlar aşağıda kısaca anlatılmıştır.

Eiji Uchino, Takeshi Yamakawa, Tsutomu Miki ve Shin Nakamura [3], gürültüsüz ve toplamsal gürültülü işaretler için bulanık mantığa dayalı basit bir aradeğerleme algoritması geliştirmişlerdir. Yapılan çalışmada ilk önce gürültüsüz işaretler için bir algoritma geliştirilmiş ve çeşitli örnekleme hızlarında giriş işareti kestirilmeye çalışılmıştır. Band genişliğinin iki katından daha düşük frekanslarda örnekleme işlemi yapıldığında kestirilen işarette bozulmalar meydana geldiği görülmüştür.

Gürültülü işaretler için de gürültüsüz duruma benzer basit bir algoritma geliştirilmiştir. İlk önce bir işarete belli zaman aralıklarıyla varyansları birbirinden farklı Gauss gürültüleri eklenmiş ve farklı örnekleme hızlarında çıkıştaki işaret gözlenmiştir. Böylece algoritmanın en iyi sonucu veren durumu elde edilmiştir. Daha sonra ise girişe farklı frekanslarda gürültüsüz bir işaret verilerek çıkış işareti gözlenmiş ve giriş-çıkış işaretine bağlı kazanç bulunmuştur. Buna bağlı olarak geliştirilen yöntemin bir alçak geçiren süzgeç gibi davrandığı görülmüştür. Bundan sonra da bu algoritma pratikteki işaretlere uygulanmıştır. Örneğin gürültülü bir ses işaretinin kestirimi yapılmıştır.

Joseph E. Chen ve Kevin N. Otto [5], bulanık mantık üyelik fonksiyonlarının örnek değerlerine bağlı olarak kestirilmesi işlemini yapmışlardır. Burada gözönünde bulundurulan gürültüsüz işaretlerdir ve üyelik fonksiyonu oldukları içinde işaretin değişim aralığı bir (1) ile sıfır (0) arasındadır. Geliştirdikleri yöntemi kübik eğri uydurma ve en küçük kareler yöntemleri ile karşılaştırmışlar ve bu yöntemlere göre geliştirilen yöntemin daha iyi sonuç verdiğini göstermişlerdir. Burada yine örnekleme frekansı band genişliğinin iki katından daha az olduğunda frekans örtüşmesinin meydana geldiği gözlenmiştir.

Sonuçta; geliştirilen yöntemle ilk yöntemin, veri sayısına göre performansları, gürültüye karşı duyarlılıkları ve nasıl bir süzgeç karakteristiği gösterdiklerine ait performansları karşılaştırılmıştır.

Kestirilen işaretler ve elde edilen sonuçlar, bulgular bölümünde verilmiştir.

Ortaya çıkan sonuçların irdelenmesi ve karşılaştırılması ise irdeleme bölümünde ele alınmıştır.

1.2. Doğal Mantık

Geleneksel mantık sisteminde yalnızca bir (1) ve sıfırlar (0) bulunur. Dolayısıyla belirsiz, kesin olmayan ya da karmaşık bir problemin çözümünde bu yöntem yetersiz kalır, hatta bazen bu yöntemle çözümler olanaksız olur. Gerçel dünya dilini kullanan bulanık mantık, bir takım dilsel niteleyiciler yardımıyla biraz sıcak, çok soğuk, hafif soğuk, çok fazla yüksek gibi günlük yaşamımızda kullandığımız kelimeler yardımıyla insan mantığına en yakın doğrulukta elektronik denetimi gerçekleştirebilir.

1.3. Tarihçe

Geliştirilmesi olanaksız matematik modellere dayalı karmaşık sistemler 1960 yılı ortalarında araştırmacılara artan bir hızda gelmeye başladı. 1965 yılında Kaliforniya Üniversitesinden Prof. Lotfi Zadeh ilk defa bulanık küme kuramının temel taşı olan "yumuşak" yaklaşım ile sistem tanıma ve tasarımını gerçekleştirdi. 1966'da Bulanık Mantık Bell Laboratuvarlarında, Dr. Peter Marinos tarafından oluşturuldu. 1972 yılında Londra Üniversitesinde Prof. E. H. Mandami bulanık mantık temelli uzman sistemle bir buhar türbininin hızının ve performansının çok başarılı bir şekilde denetlenebileceğini gösterdi. Bunun gibi birçok üniversitelerde ve firmalarda bulanık mantık üzerine çalışmalar yapılmış ve başarılı uygulamaların ardından bulanık mantığa olan ilgi artmış ve uluslararası bir çalışma ortamı oluşturabilmek amacıyla 1989 yılında LIFE (Laboratory for International Fuzzy Enginering) laboratuvarları kurulmuştur.

1.4. Bulanık Küme Kuramı

1.4.1. Geleneksel Mantık İle Karşılaştırma

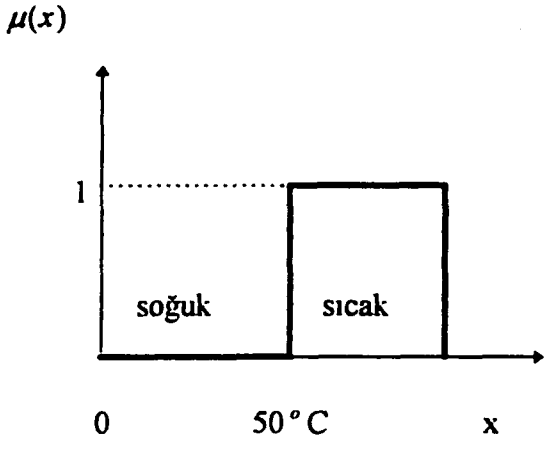
Geleneksel mantığın temelinde aslında olasılık hesapları yatar. Yani, bir olayın olabilirliği, bu mantıkla çözümlenmeye çalışılır. Sonuç ise yalnızca evet ya da hayırla sınırlıdır. Ancak bulanık mantık bundan tamamen farklıdır ve olabilirliği değil ne kadar

olduğunu verir. Dolayısıyla alınacak cevap evet ya da hayırla beraber bunların ara değerlerini de içerir [6-9].

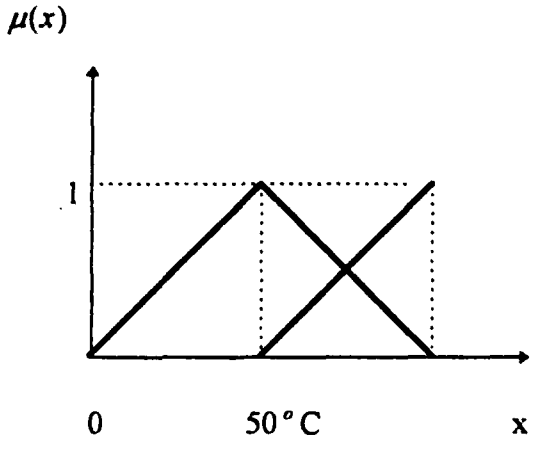
1.4.2. Bulanık ve Geleneksel Mantıkta Küme Kavramı

Bulanık mantık temel olarak insan düşünüş şeklini örnek almaktadır. Dolayısıyla bu mantığın küme elemanları da insan mantığına çok yakın ya da aynı olan elemanlardır. Örneğin bir suyun sıcaklık ayarı sıcak , çok sıcak, ılık, soğuk ya da çok soğuk gibi dilsel niteleyicilerle ifade edilebilir. İşte burada kullanılan dilsel niteleyiciler, ifade şekli ve sayısı ne olursa olsun bulanık küme elemanlarını meydana getirir. Ancak geleneksel mantıkta bu niteleyiciler ya buz gibi su (0) ya da kaynar su (1) olarak ifade edilebilir. Şekil 1.de geleneksel küme elemanlarının birbirine geçiş sürecinin ne kadar kesin olduğu görülmektedir. Örneğin $50^{o}C$ sıcaklık sınırı olarak kabul edilirse, geleneksel mantığa göre $49^{o}C$ soğuk kabul edilmesi gerekir.

Fakat bulanık küme elemanlarının birbirine geçişi yumuşaktır (Şekil 2). Eğer bu işlevi $\mu(x)$ olarak tanımlarsak, işlevin alacağı değer sıfır ile bir arasında olacaktır. Yani işlevin doğruluk değeri $\mu(x) \rightarrow (0,1)$'dir. Dolayısıyla sıcak ve soğuk arasındaki herhangi bir değer de tanımlanabilmektedir.



Şekil 1. Geleneksel mantık



Şekil 2. Bulanık mantık

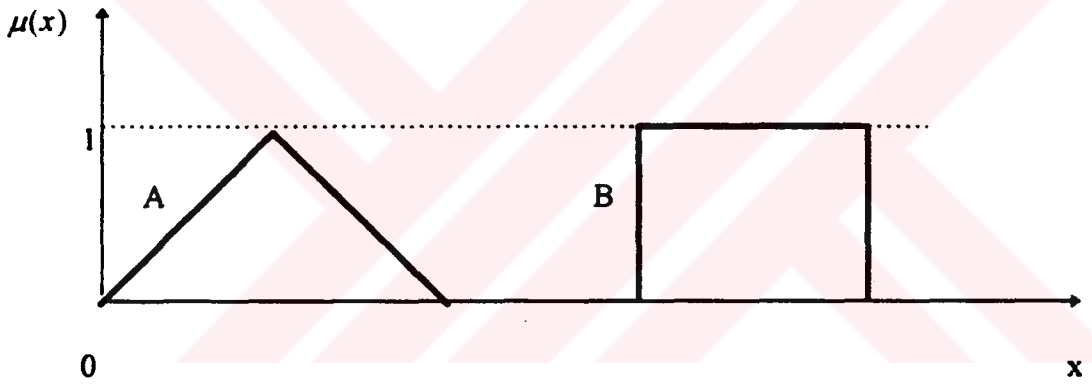
1.4.3. Bulanık ve Geleneksel Mantıkta Küme İşlemleri

Bulanık kümenin her elemanı, bu küme içinde bir üyelik değerine sahiptir ve bulanık A kümesinin işlev haritası 0 ile 1 arasındaki gerçel sayılardan oluşur. (Şekil 3) Yani $\mu(x) \in [0,1]$ olur.

X evrensel kümesinde yer alan, sınırlı ve sonlu sayıdaki x elemanlarından oluşan A bulanık kümesi,

$$A = \{\frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_1)}{x_2} + ...\} = \{\sum_i \frac{\mu_A(x_i)}{x_i}\} = \{\int \frac{\mu_A(x)}{x}\} \qquad (1)$$

olarak ifade edilebilir.



Şekil 3. Bulanık küme üyelik fonksiyonları

X evrensel kümesinde A, B ve C bulanık kümeleri tanımlayalım ve x yine bu kümelere ait bir eleman olsun :

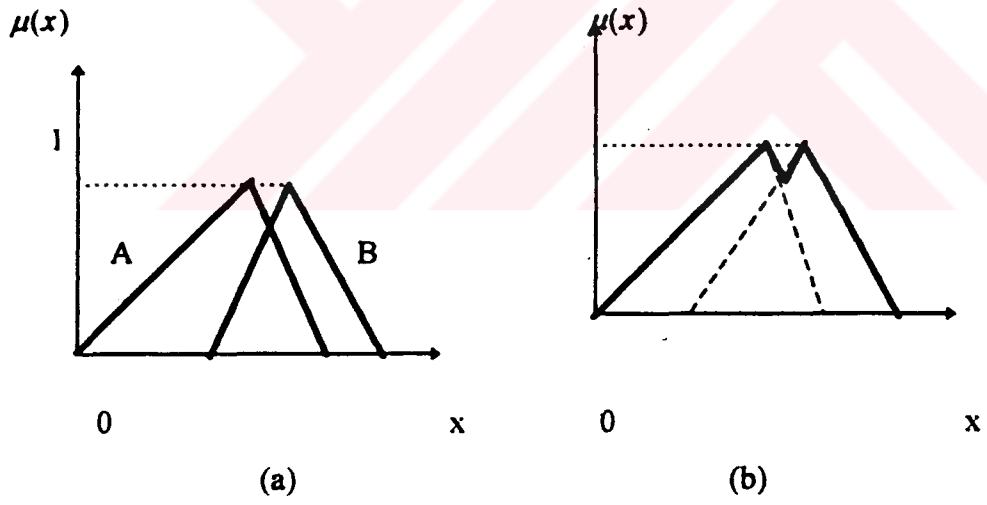
$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) \qquad (2)$$

$$\mu_{A \cap C}(x) = \mu_A(x) \wedge \mu_C(x) \qquad (3)$$

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \qquad (4)$$

eşitlikleri yazılabilir. Bu ifadelere ait grafikler şekil 4 , 5 ve 6' da gösterilmiştir.

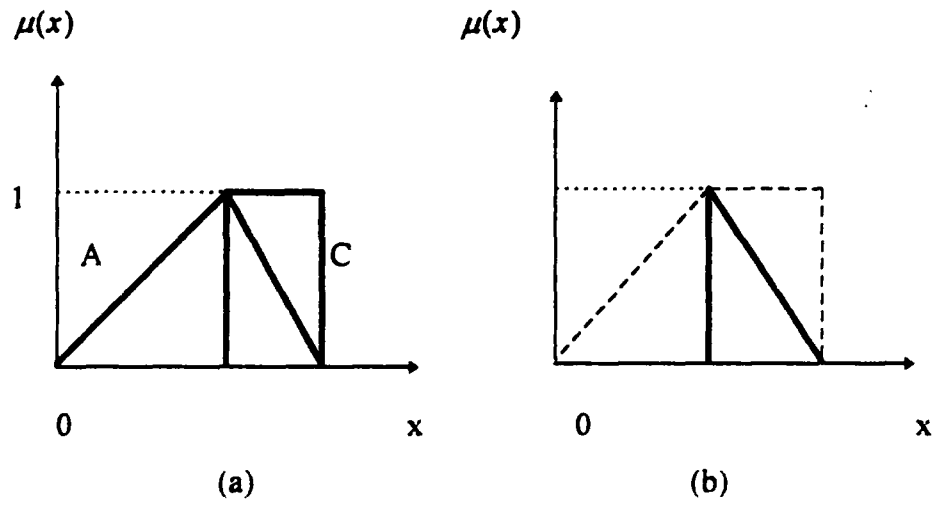
Örneğin A bulanık kümesi, A={ $\frac{1}{2}+\frac{0.3}{4}+\frac{0.4}{5}$ } ve B bulanık kümesi de, B={ $\frac{0.1}{2}+\frac{0.4}{4}+\frac{0.6}{5}$ } olsun. A bulanık kümesinin evriği $\overline{A}=\{\frac{1}{1}+\frac{0}{2}+\frac{1}{3}+\frac{0.7}{4}+\frac{0.6}{5}\}$ ve B bulanık kümesinin evriği $\overline{B}=\{\frac{1}{1}+\frac{0.9}{2}+\frac{1}{3}+\frac{0.6}{4}+\frac{0.4}{5}\}$ olur. A ve B bulanık kümelerinin birleşimi $A\cup B=\{\frac{1}{2}+\frac{0.4}{4}+\frac{0.6}{5}\}$ ve A ve B bulanık kümelerinin kesişimi ise $A\cap B=\{\frac{0.1}{2}+\frac{0.3}{4}+\frac{0.4}{5}\}$ olur. Görüldüğü gibi bulanık kümelerdeki işlemler geleneksel kümelerdeki işlemlerle benzeşmektedir. Ancak bulanık kümelerdeki evirme işlemine ait özelliklerde bazı farklılıklar vardır. A bulanık kümesinin evriği ile birleşimi evrensel küme değildir ve A bulanık kümesinin evriği ile kesişimi boş küme değildir. Matematiksel olarak : $A\cup\overline{A}\neq X$ ve $A\cap\overline{A}\neq 0$ dır.(X= Evrensel küme)



Şekil 4. Küme işlemleri

a) A ve B kümeleri

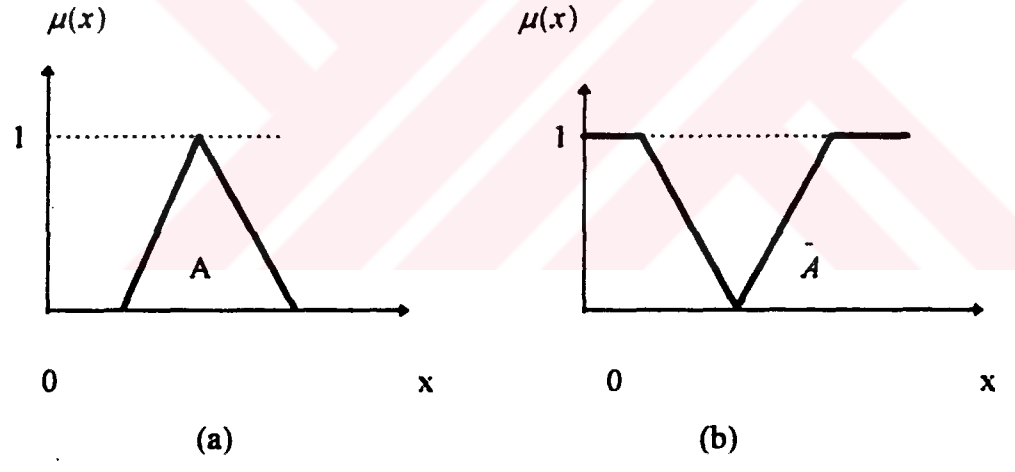
b) A ve B kümelerinin birleşimi



Şekil 5. Küme işlemleri

a) A ve C kümeleri

b) A ve C kümelerinin kesişimi



Şekil 6. Küme işlemleri

a) A kümesi

b) A kümesinin tümleyeni

Bunun dışındaki diğer mantıksal işlem özellikleri, geleneksel mantıktaki işlem özellikleriyle benzeşirler.

$$A \cup (B \cup C) = (A \cup B) \cup C \quad (5)$$

$$A \cap (B \cap C) = (A \cap B) \cap C \quad (6)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad (7)$$

$$(A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad (8)$$

$$A \cup A = A \text{ ve } A \cap A = A \quad (9)$$

$$A \cup \varnothing = A \text{ ve } A \cap X = A \quad (10)$$

$$A \cap \varnothing = \varnothing \text{ ve } A \cup X = A \quad (11)$$

Eğer

$$A \subseteq B \subseteq C \quad (12)$$

ise,

$$A \subseteq C \quad (13)$$

olur. (X = Evrensel küme)

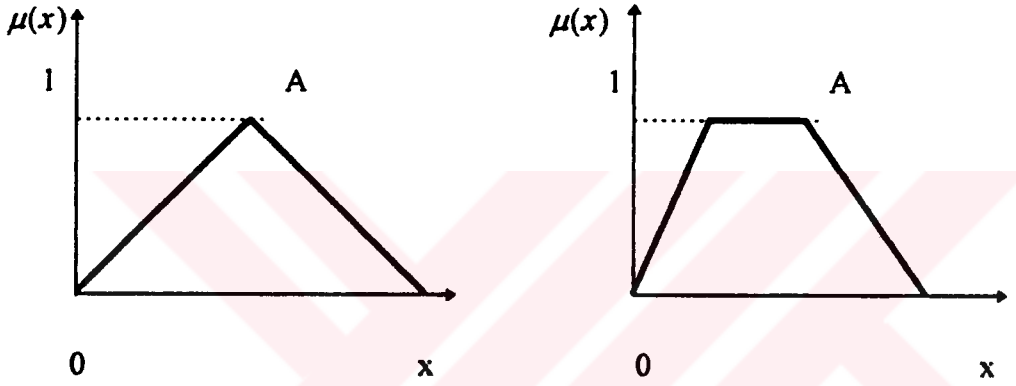
1.4.4. Üyelik Fonksiyonları

Bulanık mantık süreçlerinin başlatılabilmesi için gerekli üyelik işlevleri, dilsel niteleyicilerden oluşan bir anlam gurubudur. Bu anlam üyelik işlevlerinin ağırlık merkeziyle tanımlanır. Üyelik işlevleri genellikle biçimsel olarak gösterilir . En çok kullanılan üyelik fonksiyonları üçgen, yamuk veya çan eğrisidir. (Şekil 7)

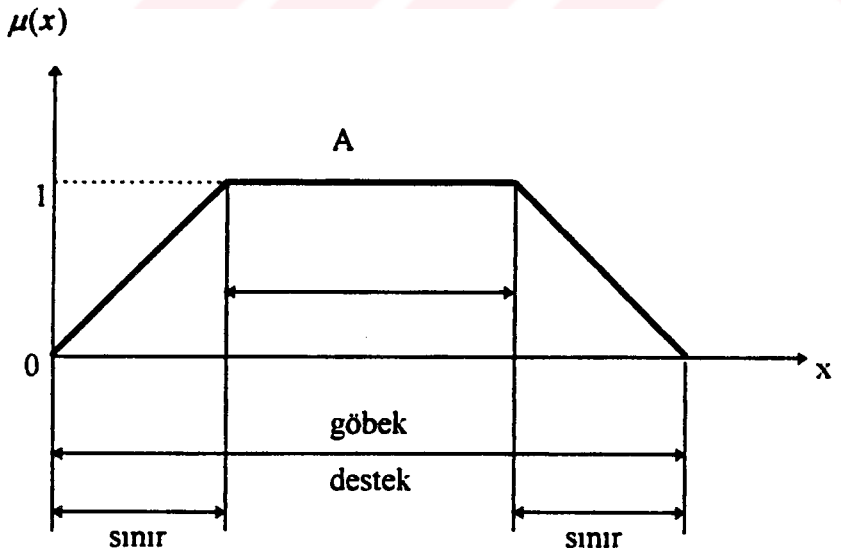
1.4.5. Üyelik İşlevlerinin Özellikleri

Bulanık kümenin grafik olarak gösteriminde daha da kolay görülebileceği gibi bazı üyelik işlev tanımları bulunmaktadır. (Şekil 8)

Bulanık kümeye ait üyelik işlevinin $\mu_A(\chi) = 1$ olan bütün x elemanlarını kapsayan bölgesi göbek olarak adlandırılır. Yine bulanık kümeye ait üyelik işlevinin $0 < \mu_A(\chi) < 1$ olan bütün x elemanlarını kapsayan bölgesi sınır olarak adlandırılır. Bulanık kümeye ait $\mu_A(\chi) > 0$ olduğu bütün bölgeler ise destek bölgesidir.



Şekil 7. Bulanık üye fonksiyonları



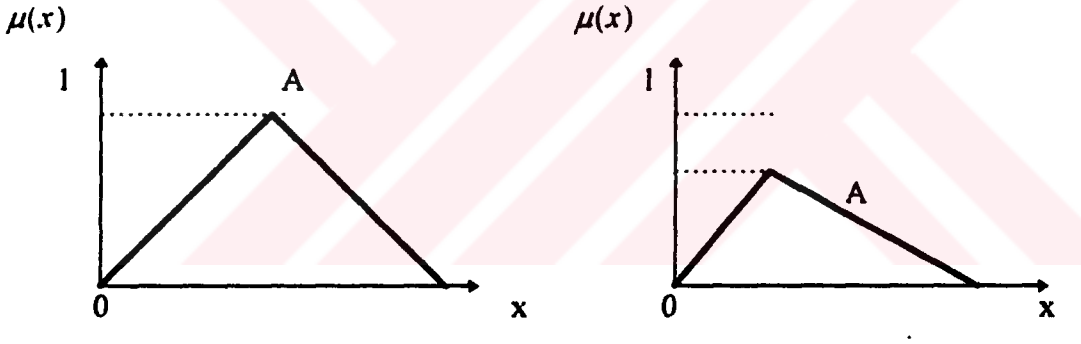
Şekil 8. Üyelik fonksiyonunun özellikleri

Olağan (normalize) bulanık küme, üyelik değerinin en az bir x elemanı için 1 değerini aldığı bulanık kümedir. Eğer bulanık kümede bir ve yalnızca bir elemanın üyelik değeri 1 ise, bu eleman, kümenin prototipi ya da prototip elemanı olarak adlandırılır. Bunun dışındakilere kümelere olağanaltı bulanık kümeler denir. (Şekil 9)

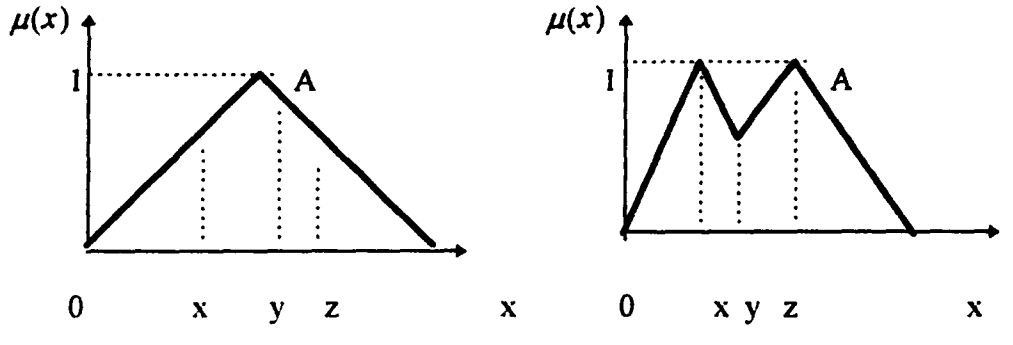
Bulanık kümenin üyelik işlenin üyelik değerleri monoton artan ve daha sonra monoton azalan bir durumda ise ya da belli üyelik değerlerinde 1 olduktan sonra monoton azalan ise böyle kümelere bulanık dışbükey kümeler denir. Başka bir deyişle x, y, z elemanları A bulanık kümesinin içinde olsun, $x < y < z$ olmak şartıyla

$$\mu_A(x) \geq \min[\mu_A(y), \mu_A(z)] \qquad (14)$$

denklemini sağlayan bulanık kümeler dışbükeydir. Bu klasik matematiksel dışbükey biçim tanımından farklıdır. (Şekil 10)



Şekil 9. Olağan ve olağanaltı bulanık kümeler



Şekil 10. Olağan, dışbükey olan ve dışbükey olmayan bulanık küme üyelik fonksiyonları

1.4.6. Bulanıklaştırma

Fiziksel giriş bilgilerinin, dilsel niteleyicilerle ifade edebileceğimiz bulanık mantık bilgileri şekline çevirme işlemine bulanıklaştırma adı verilir. Ancak bu bilgilerin tamamının mutlaka kesin bilgiler olması söz konusu değildir. Bulanıklaştırma işlemi önemli ölçüde kesin olmayan bilgiyi de içine alır ve bulanıklaştırır. Bulanıklaştırma sonucu elde edilen değişkenlere dilsel değişkenler denir ve işlemle tüm giriş değişkenlerinin değerleri, üyelik derecesi olarak buraya atanır. Örneğin , 100 kmlik bir uzaklık giriş bilgisi, dilsel niteleyici olarak "uzak" olarak ifade edilebilir. Bununla beraber yine 100-150 km arası, tam kesin olmayan bir bilgi olarak yine "uzak" olarak ifade edilebilir.

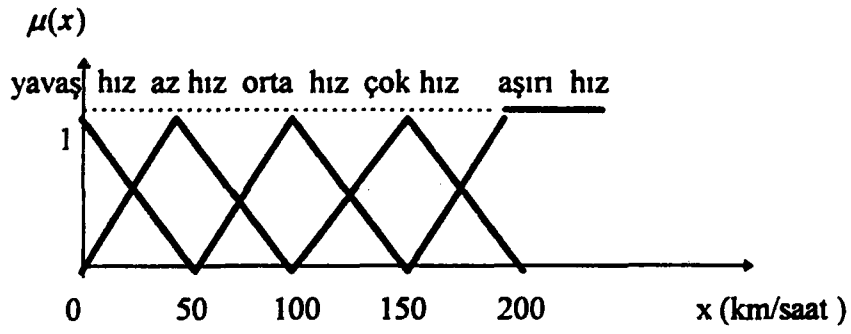
Bulanıklaştırma işlemi göreceli olarak bu kadar kolay olmasına karşın, uzman sistem kalıplarından dolayı bu işlemlerin yapılması büyük ölçüde deneyime dayanmaktadır. Operatörün sistemde çalışırken gösterdiği davranışlar, sistemin matematiksel modelinden daha önemlidir. Dolayısıyla bulanıklaştırma aşamasına gelinebilmesi için gerekli süre bazen çok uzun olabilir. Bununla birlikte kesin olmayan bilgileri kullanabilmesi, sürecin matematiksel modeline gereksinim duyulmaması ve uygulamaya çabucak geçilebilmesi, bütün bunlardan sonra da yüksek derecede verim alınabilmesi bulanık mantığın önemini açıkça ortaya koymaktadır.

1.4.7. Dilsel Değişkenler

Bunu daha iyi açıklayabilmek için bir örnek ele alalım: Yolda giden bir arabanın sürücüsü önüne çıkan engel karşısında nasıl hareket edebilir?

Buradaki girdilerimiz, hız ve uzaklık ; buna bağlı olarak elde edeceğimiz çıktımız ise fren olsun.

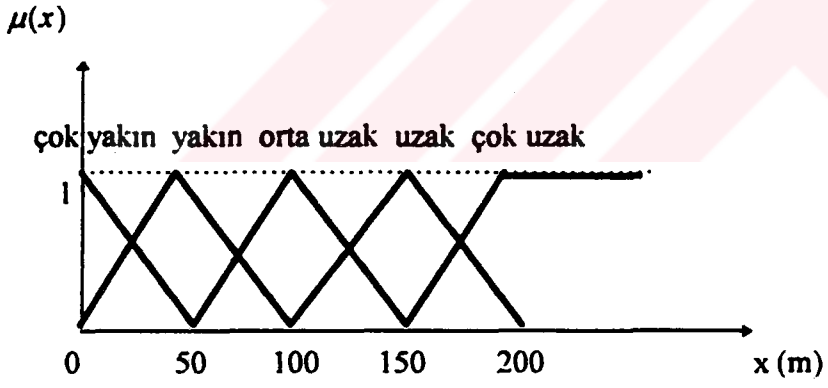
Bulanık kümeler genellikle üç, beş ya da yedi üyelik fonksiyonlarından oluşabilir. Örneğimize göre; yavaş, az, orta, çok ve aşırı hızlı şeklinde beş üyelik fonksiyonuna sahip bir bulanık hız kümesi oluşturabiliriz. (Şekil 11)



Şekil 11. Hız için üyelik fonksiyonları

Burada da görüldüğü gibi tanımlar tamamıyla insanların söylemlerine göre geliştirilmiştir ve dilsel niteleyiciler olarak anılırlar. Bunların işlevsel olarak elde edilmeleri ve uygulama aşamasına getirilmeleri büyük ölçüde sistemde daha önce elde edilmiş deneyimlere bağlıdır ve biz bu sistemlere uzman sistemler adını vermekteyiz.

Yine aynı şekilde uzaklık kavramına ilişkin bir kümeyi de; çok uzak , orta uzak, yakın ve çok yakın şeklinde belirleyebiliriz.(Şekil 12)



Şekil 12. Uzaklık için üyelik fonksiyonları

Burada dikkat edilmesi gereken bir başka nokta ise, örneğimizdeki bulanık kümelerin olağan dışbükey bulanık kümeler olduğudur.

1.4.8. Bulanık Çıkarım

Bulanık mantıkta da geleneksel mantıkta olduğu gibi bazı mantık işlemleri yer almaktadır. Ancak bu işlemlerin komutları VE, VEYA, DEĞİL , EĞER ve İSE (ÖYLEYSE) ile sınırlı çok basit ve ayni zamanda da kullanışlıdır. Bu kurallar bütününe kurallar ya da bulanık mantık denetleyicisi üzerinde kural tabanı denir. Şimdi bulanık mantık ile girdilere göre çıktı değerini inceleyelim.

Kural 1: EĞER araba çok hızlı VE engel orta uzaklıkta İSE frene çok kuvvetle basınız.

Kural 2: EĞER araba aşırı hızlı VE engel çok yakında İSE frene tam kuvvetle basınız.

Kural 3: EĞER araba orta hızlı VE engel uzakta İSE frene az kuvvetle basınız.

Kural 4: EĞER araba yavaş hızlı VE engel çok uzakta İSE frene çok az kuvvetle basınız.

Hızı kısaca "h" ve uzaklığı da kısaca "u" ile gösterirsek, fren "f" fonksiyonu f(h,u) şeklinde olacaktır.

Eğer değişkenler arasında VE kullanılmış ise buna bağlı olarak ortaya çıkacak fonksiyon minimum değer alacaktır. Yani, f((h) VE (u)) = min (f(h),f(u)) dır.

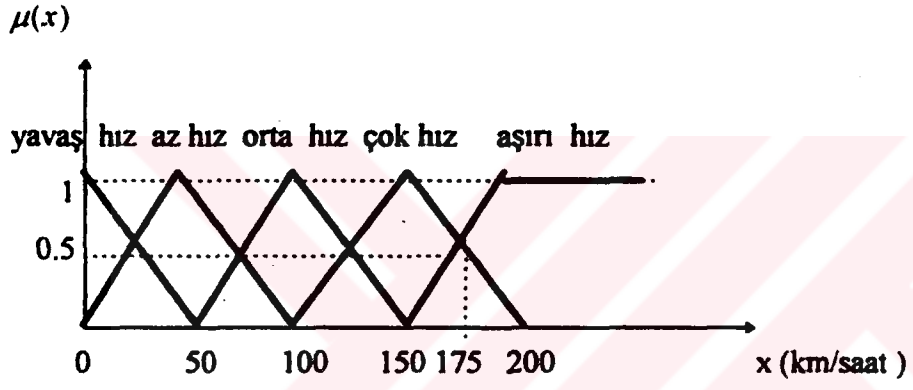
Değişkenler arasında kullanılan şart bağlacı VEYA ise fren fonksiyonu, f((h) VEYA (u)) = max(f(h),f(u)) olacaktır.

Fonksiyonlarda kullanılan DEĞİL işlemi ise, f(DEĞİL (h)) = 1 - f(h) ya da f(DEĞİL (u)) = 1 - f(u) anlamına gelmektedir. Burada tüm f değerleri $0 \leq f \leq 1$ şeklinde olacaktır.

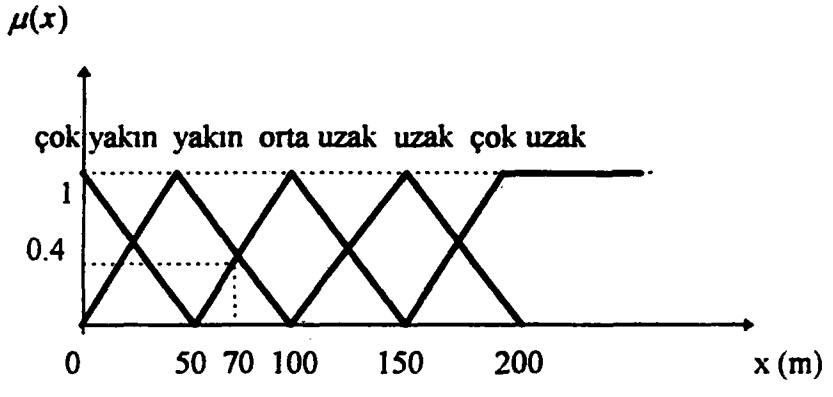
Örneğe uygun olarak (Şekil 13.a), b)) kural 1 göz önüne alınsın. Giriş bilgileri araba "çok hızlı" ve uzaklık ise "orta uzaklık" verilmişti ve buna bağlı olarak çıkarım "frene orta kuvvette bas"tır. Şekil 14'de bu girişlere ilişkin çıkarım değerleri grafiksel olarak verilmiştir. Değişkenler arasında VE şart bağlacı kullanıldığı için burada elde edilen iki doğruluk değerinden (0. 5 , 0.4) küçük olan seçilmiştir. Yani elde edilen çıkarım %60 fren şiddetidir.

Burada dikkat edilmesi gereken önemli bir nokta, bulanık mantıkta, bizim geleneksel mantıkta alıştığımız gibi bir çıkarım yapılmamasıdır.

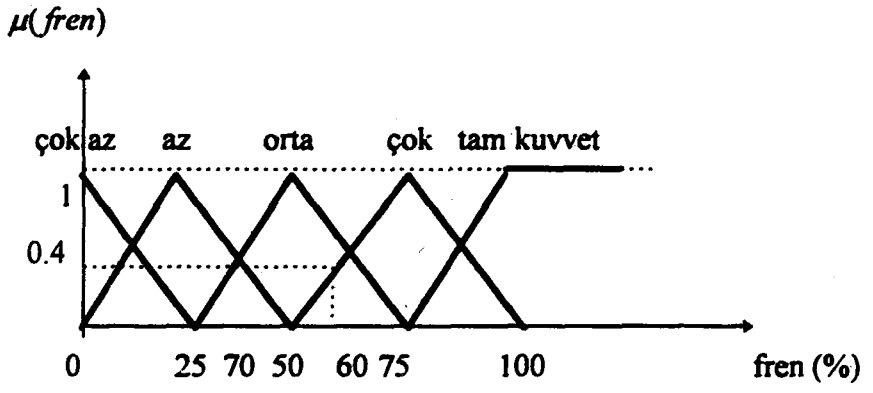
Doğrusal olmayan sistemlerde kullanılan bulanık çağrışımlı eşleşme ya da bellek çıkarım tekniği de bulunmaktadır. Bu teknikte her bir kural için, o kuralın tek başına önemliliğini ortaya koyan destekleme derecesinin de belirtildiği tablolar oluşturulur. Burada min-max yönteminde olduğu gibi yalnızca 0 ve 1 değerlerinde değil, destekleme derecesi 0 ile 1 arasındaki değerler de alınabilir. Ancak bir sonuç için birden fazla kural oluşturulmuş ise, bunların maksimum dereceli olanı, bütün hepsi için destekleme derecesi olarak seçilir.



Şekil 13. a) Hızı 175 km/saat olan arabanın üyelik fonksiyonundan elde edilen girdi çıkarımı



Şekil 13. b) Engelin arabaya uzaklığı ile ilgili üyelik fonksiyonundan elde edilen girdi çıkarımı



Şekil 14. Frene basma şiddeti

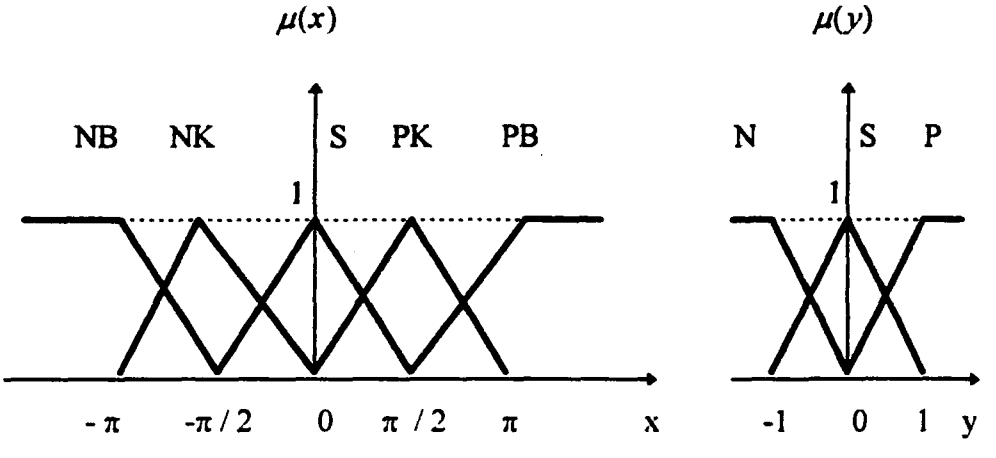
Bir başka örnek daha verelim. y = cos(x) doğrusal olmayan bir fonksiyonu için, y çıkışını yaklaşık olarak bulan bulanık kurallarını gerçekleyelim. [- π, π] aralığında şekil 15'de gösterildiği gibi, x girişi için 5 üyelik fonksiyonu ve [-1, 1] aralığında y çıkışı içinse 3 üyelik fonksiyonu alınmıştır. Girişe bağlı olarak çıkışları aşağıda belirtilen 4 bulanık kuralla bulabiliriz:

Kural 1: EĞER x = S İSE y = P olur.

Kural 2: EĞER x = NK VEYA PK ise y = S olur.

Kural 3: EĞER x = NB İSE y = N olur.

Kural 4: EĞER x = PB İSE y = N olur.



Şekil 15. y = cos(x) fonksiyonu için giriş ve çıkış üyelik fonksiyonları

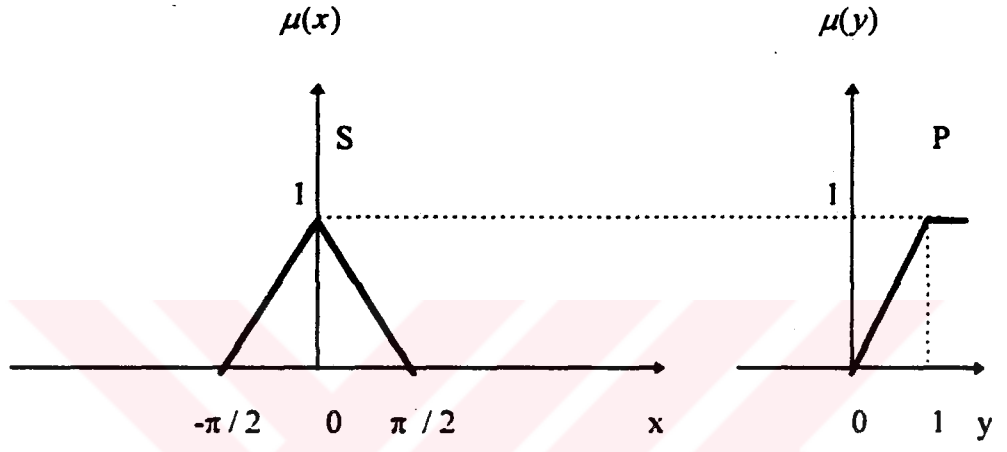
x = 0 ise, kural 1'den y = 1 bulunur. (Şekil 16. a))

x = π / 2 ise, kural 2'den y = 0 bulunur. (Şekil 16. b))

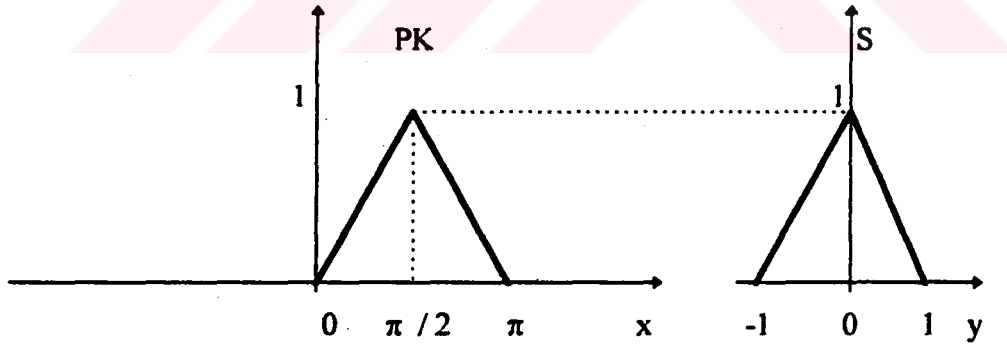
x = π ise, kural 4'den y = -1 bulunur.

x = -π ise, kural 3'den y = -1 bulunur.

x = -π / 2 ise, kural 2'den y = 0 bulunur.



(a)



(b)

Şekil 16. x giriş değerleri için y çıkış değerlerinin üyelik fonksiyonlar kullanılarak bulunması

a) x = 0 ise, kural 1'den y = 1 olur

b) x = π / 2 ise, kural 2'den y = 0 olur

2. YAPILAN ÇALIŞMALAR

Pratik sistemlerin karakteristiklerinden biri, neden-sonuç veya etki-tepki çiftiyle tanımlanabilir. Sistem mühendisliği alanında neden ve sonuç arasındaki ilişki bir giriş-çıkış fonksiyonu ile tanımlanabilir. Bu veri çiftleri ne kadar fazla olursa sistemin işlevinin de daha iyi anlaşılacağı ortadadır. Yani, giriş-çıkış fonksiyonu, giriş-çıkış verisi dinamik bölge içindeki her nokta için alınarak tamamen belirlenebilir. Fakat bu işlem çok zaman ve işlem gerektirmektedir. Eğer bu fonksiyon az miktardaki veriden yaklaşık olarak belirlenebilirse, veri azaltmaya olanak verir, böylece zaman ve işlem sayısı azalabilir. Örneklenmiş bir veri bir pratik sistemin girişine verildiğinde gürültü nedeniyle bozulabilir ve azalabilir. Bu durumda gözlenen veri değerlerini birleştiren sürekli bir işaretin bulunabilmesi için bir aradeğerleme algoritması yetersiz kalmaktadır. Çünkü bu işlemde kullanılacak veriler gerçek değerleri göstermemektedir.

Bu çalışmada gürültülü ve eksiltilmiş veriler için bulanık mantığa dayalı bir aradeğerleme algoritması geliştirilmiştir ve bu algoritma hem gürültülü hem de gürültüsüz işaretler için uygulanmıştır. Geliştirilen yöntemdeki bulanık süzgeç bir alçak geçiren süzgeçtir.

2.1. Gürültüsüz Verilerin Bualnık Mantığa Bağlı Aradeğerlemesi

Ayrık, gürültüsüz veriler için bulanık mantık kuralına bağlı bir aradeğerleme algoritması verilmiştir. Bunun için giriş-çıkış ifadesi y = f(x) şeklinde bir sistemi ele alalım. Burada f(x) giriş-çıkış fonksiyonu, uygulanan girişler için çıkışlar ölçülerek belirlenebilir. Giriş-çıkış verilerini de (x,y) çiftleriyle gösterelim. Burada amaç, giriş-çıkış fonksiyonunu mümkün olduğu kadar az bulanık mantık verisi kullanarak tanımlamaktır.

Ölçülen giriş-çıkış veri çiftlerinin $\{(x_i,y_i)|i = 1...,n\}$ şeklinde oluşturduğu kümenin 2 boyutlu Euclidean uzayında bir nokta olduğu düşünülmektedir. Buradaki (x_i,y_i) ölçülen giriş- çıkış veri çiftlerini göstermektedir. İnceleme tek değişkenli durum için yapılmıştır.

Şekil 17 'de gösterildiği gibi $[x_i,x_{i+1}]$ aralığında bir aradeğerleme işlemi ele alalım. $[x_{i-1},x_i]$ ve $[x_i,x_{i+1}]$ komşu bitişik aralıkları göstermektedir. $f_i(x)$ ardışıl iki nokta olan $[x_{i-1},y_{i-1}]$ ve $[x_i,y_i]$'den ve $f_{i+1}(x)$ de $[x_{i-1},y_{i-1}]$ ve $[x_{i+1},y_{i+1}]$ noktalarından geçen iki doğru parçasıdır ve $x_{i-1} < x_i < x_{i+1} < x_{i+2}$ 'dir. $[x_i,x_{i+1}]$ aralığında önerilen aradeğerleme algoritması basit olarak aşağıdaki gibi tanımlanmaktadır:

a) $f_i(x)$ ve $f_{i+1}(x)$ fonksiyonlarına sırasıyla $\mu_i(x)$ ve $\mu_{i+1}(x)$ üyelik fonksiyonları atanmalıdır.

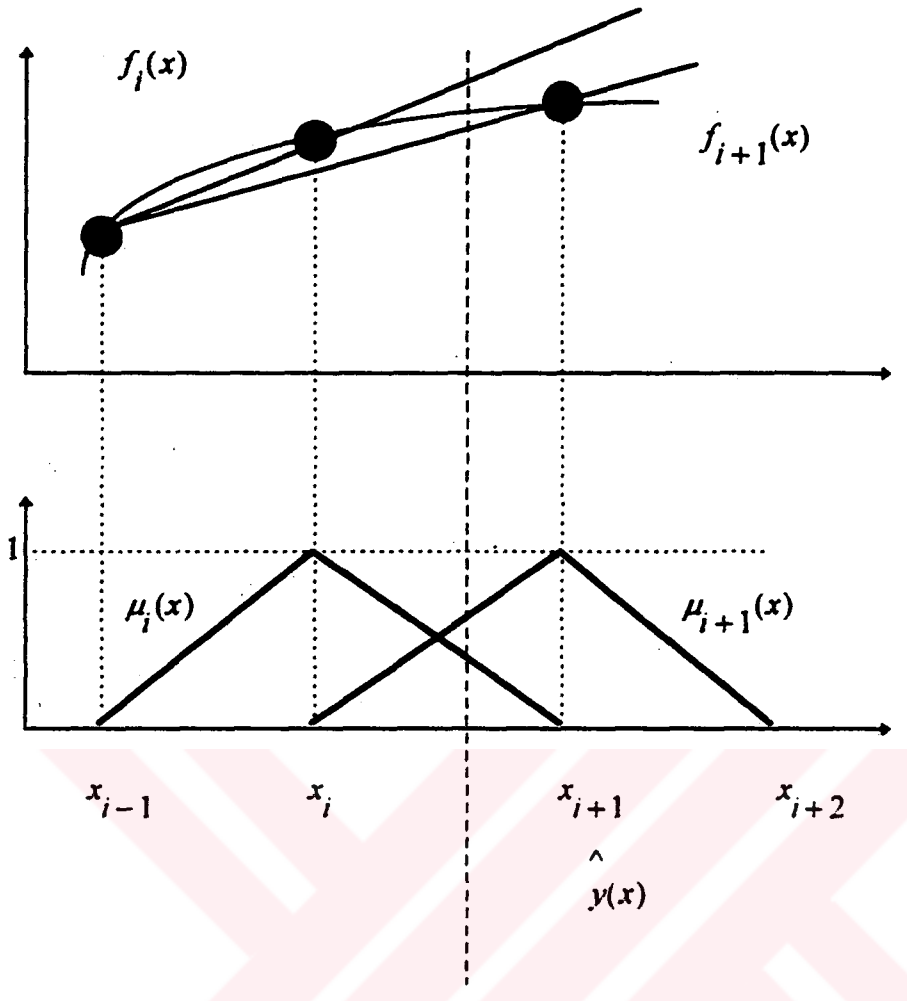
b) $[x_i,x_{i+1}]$ aralığındaki aradeğerleme aşağıdaki kurala göre yapılmaktadır:

Burada $A_j(j=i-1,i,i+1)$, üyelik fonksiyonları $\mu_j(j=i-1,i,i+1)$ olan bir bulanık kümesidir. u , ilgili fonksiyona atanan üyelik fonksiyonunu göstermektedir. Aradeğerlenmiş $\hat{y}(x)$ değeri aşağıdaki gibi bulunmaktadır:

$$\hat{y}(x) = \frac{\mu_i(x)f_i(x) + \mu_{i+1}(x)f_{i+1}(x)}{\mu_i(x) + \mu_{i+1}(x)} \qquad (15)$$

Kural 1: Eğer $x = A_i$ ise $u = f_i(x)$ 'dır.

Kural 2: Eğer $x = A_{i+1}$ ise $u = f_{i+1}(x)$ 'dır.



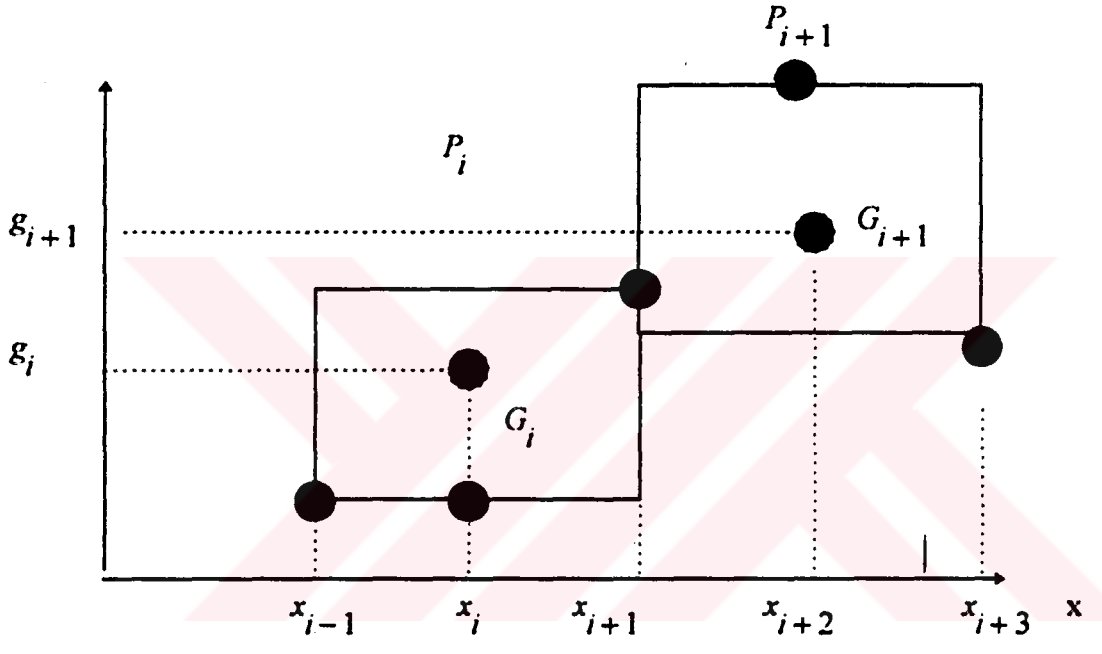
Şekil 17. $f_i(x)$ ve $f_{i+1}(x)$ fonksiyonlarına atanan üyelik fonksiyonları

2.2. Ayrık Rastlantı İşaretlerinin Bulanık Mantığa Dayalı Aradeğerlemesi

Örneklenmiş işaret ve gürültü, çoğu durumlarda durağan olmadığından işaret/ gürültü oranı devamlı değişir. Ayrıca, işaret ve gürültünün dağılım özellikleri çoğunlukla Gauss değildir. Gürültü azaltmada Kalman süzgeçleme gibi süzgeçleme yöntemleri iyi sonuç verir. Ancak bu yöntemlerin uygulanmasında işaretin dinamiğinin ve istatistiğinin önceden bilinmesi ve bazı kısıtlarda (örneğin Gauss tipi) olması gerekmektedir. Bu idealize şartların aşılması için birçok çalışma yapılmıştır. Fakat bu çalışmalarda kullanılan algoritmaların çoğu karmaşıktır ve fazla işlem gerektirir. Burada sistem dinamiğinin bilinmesini gerektirmeyen bir algoritma üzerinde durulacaktır. Sistem çıkışındaki işaretin y(x) = k(x) + v(x) şeklinde olduğu varsayılsın.

Burada k(x) zaman domeninde bir işaret, v(x) ise istatistiği bilinmeyen toplamsal gürültüdür ve y(x) ise gürültü ile bozulmuş çıkış işaretidir. Buradaki amaç, y(x) değerlerini kullanarak bir bulanık mantık yardımıyla k(x) giriş işaretini kestirmektir.

Aradeğerleme prensibi, gürültü ile bozulmuş işaretten yararlanarak gürültüsüz işareti kabaca tahmin etmektir. Önerilen aradeğerleme algoritmasında önce hedeflenen işaretin ana hatları tahmin edilir ve gürültüsüz işarette uygulanan metoda benzer işlem dizisi uygulanır. Bu algoritma ise aşağıdaki gibidir:

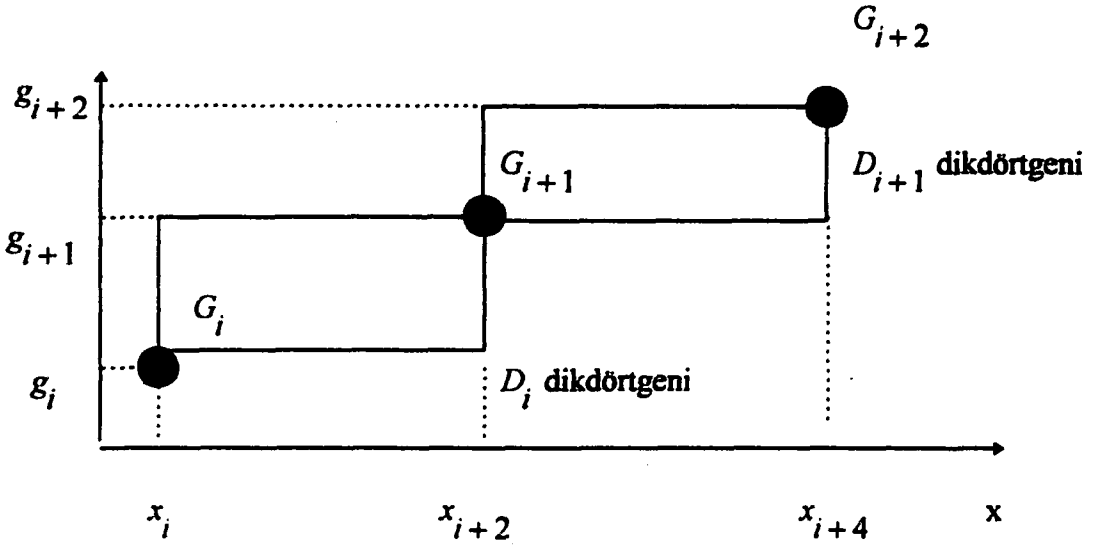


Şekil 18. Örneklenmiş gürültülü işaretin örnek değerlerinin bir dikdörtgen içerisine alınması

a) Örneklenmiş gürültülü y(x) işareti parçalara bölünür. Her parçadaki maksimum ve minimum değerler dikdörtgensel parçanın alt ve üst kenarlarında olacak şekilde bir dikdörtgenin içine alınır. Şekil 18'da dikdörtgensel parça içine alma işlemi gösterilmiştir. Her parça 3 örnek noktadan oluşmaktadır.

b) $P_i (i = 1,...,n)$ bir dikdörtgensel parça olsun ve G_i, P_i'in ağırlık merkezi olsun.

c) G_i ve G_{i+1} ağırlık merkezleri, bir dikdörtgensel parçanın alt ve üst köşeleri olacak şekilde bir başka dikdörtgen oluşturulur. (Şekil 19)



Şekil 19. P_i Dikdörtgensel parçaların ağırlık merkezleri kullanılarak tekrar oluşturulan dikdörtgenler

d) Her G_i ağırlık merkezlerinden geçen m_i eğimin belirlenmesi gerekmektedir. Bunun için ilk önce s_i ve s_{i+1} değerleri bulunur. Çünkü s_i ve s_{i+1}'in aldıkları değerlere göre m_i değeri de değişmektedir.

$$s_i = (g_{i+1} - g_i)/(x_{i+2} - x_i) \tag{16}$$

$$s_{i+1} = (g_{i+2} - g_{i+1})/(x_{i+4} - x_{i+2}) \tag{17}$$

e) Eğer $s_i\, s_{i+1} \leq 0$ ise, $m_i = 0$ olur. Bu durum o noktanın bir yerel uç noktası olduğunu gösterir.

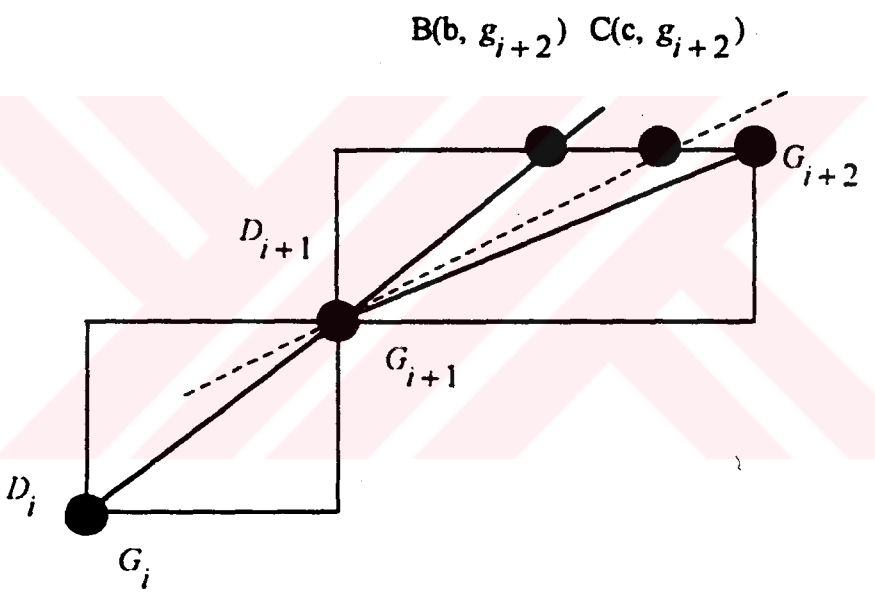
f) Eğer $|s_i| > |s_{i+1}| > 0$ ise; s_i eğimi ile G_{i+1} noktasından geçerek D_{i+1} dikdörtgeni B (b, g_{i+2}) noktasında kesen bir doğru elde edelim.

$$c = \frac{b + x_{i+4}}{2} \tag{18}$$

olsun. Bu durumda,

$$m_i = \frac{g_{i+2} - g_{i+1}}{c - x_{i+2}} \tag{19}$$

olur. Dikkat edilirse $c > \frac{x_{i+2} + x_{i+4}}{2}$ olduğu görülür. (Şekil 20)



Şekil 20. m_i eğiminin belirlenmesi

g) Eğer $0 < |s_i| < |s_{i+1}|$ ise ; s_{i+1} eğimi ile G_i noktasından geçerek D_i dikdörtgeni B (b, g_i) noktasında kesen bir doğru elde edelim.

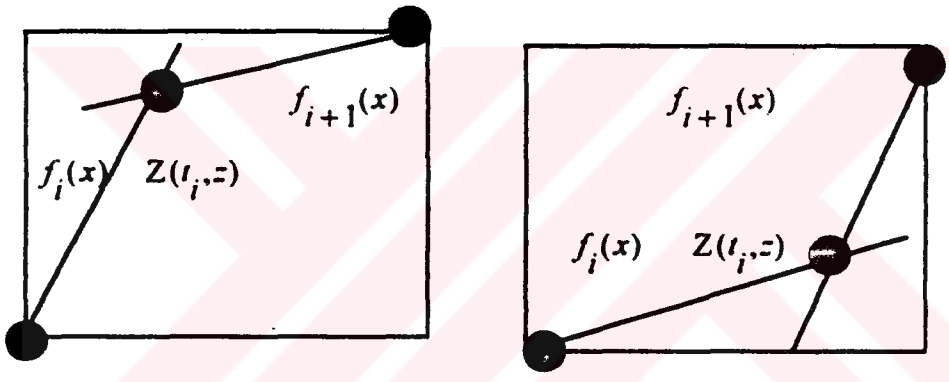
$$c = \frac{b + x_i}{2} \tag{20}$$

olsun. Bu durumda

$$m_i = \frac{g_{i+1} - g_i}{x_{i+2} - c} \tag{21}$$

olur.

h) $f_i(x)$ doğrusu m_i eğimi ile (x_{i+2}, g_{i+1}) noktasından, $f_{i+1}(x)$ doğrusu m_{i+1} ile (x_{i+4}, g_{i+2}) noktasından geçsin. $f_i(x)$ ve $f_{i+1}(x)$ doğruları D_{i+1} dikdörtgeninin içerisinde $Z(t_i, z)$ noktasında kesişsinler.(Şekil 21)



Şekil 21. $f_i(x)$ ve $f_{i+1}(x)$ doğrularının D_{i+1} dikdörtgeninin içerisinde kesişmesi durumları

Bu durumda, şekil 22'de görüldüğü gibi $V(\frac{x_{i+2}+t_i}{2}, g_{i+1} + f_i(\frac{x_{i+2}+t_i}{2}))$ ve $W(\frac{x_{i+4}+t_i}{2}, g_{i+2} + f_{i+1}(\frac{x_{i+4}+t_i}{2}))$ noktalarından geçen $f_{i+2}(x)$, G_i ve $Y(t_i, f_{i+2}(t_i))$ noktalarından geçen $f_{i+3}(x)$, $Y(t_i, f_{i+2}(t_i))$ ve G_{i+1} noktalarından geçen $f_{i+4}(x)$ doğruları elde edilir.

$f_i(x)$, $f_{i+1}(x)$, $f_{i+2}(x)$, $f_{i+3}(x)$ ve $f_{i+4}(x)$ fonksiyonlarına ait $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları bulunur. (Şekil 22)

$[x_{i+2}, x_{i+4}]$ aralığındaki aradeğerleme aşağıdaki kurala göre yapılmaktadır.

Aradeğerlenmiş $\hat{k}(x)$ değeri aşağıdaki gibi bulunmaktadır:

Kural 1: Eğer $x_{i+2} \le x < \frac{x_{i+2} + t_i}{2}$ ise,

$$\hat{k}(x) = \frac{f_{i+3}(x)\mu_i(x) + f_{i+2}(x)\mu_{i+1}(x)}{\mu_i(x) + \mu_{i+1}(x)} \tag{22}$$

Kural 2: Eğer $\frac{x_{i+2} + t_i}{2} \le x < t_i$ ise,

$$\hat{k}(x) = \frac{f_i(x)\mu_{i+1}(x) + f_{i+3}(x)\mu_{i+2}(x)}{\mu_{i+1}(x) + \mu_{i+2}(x)} \tag{23}$$

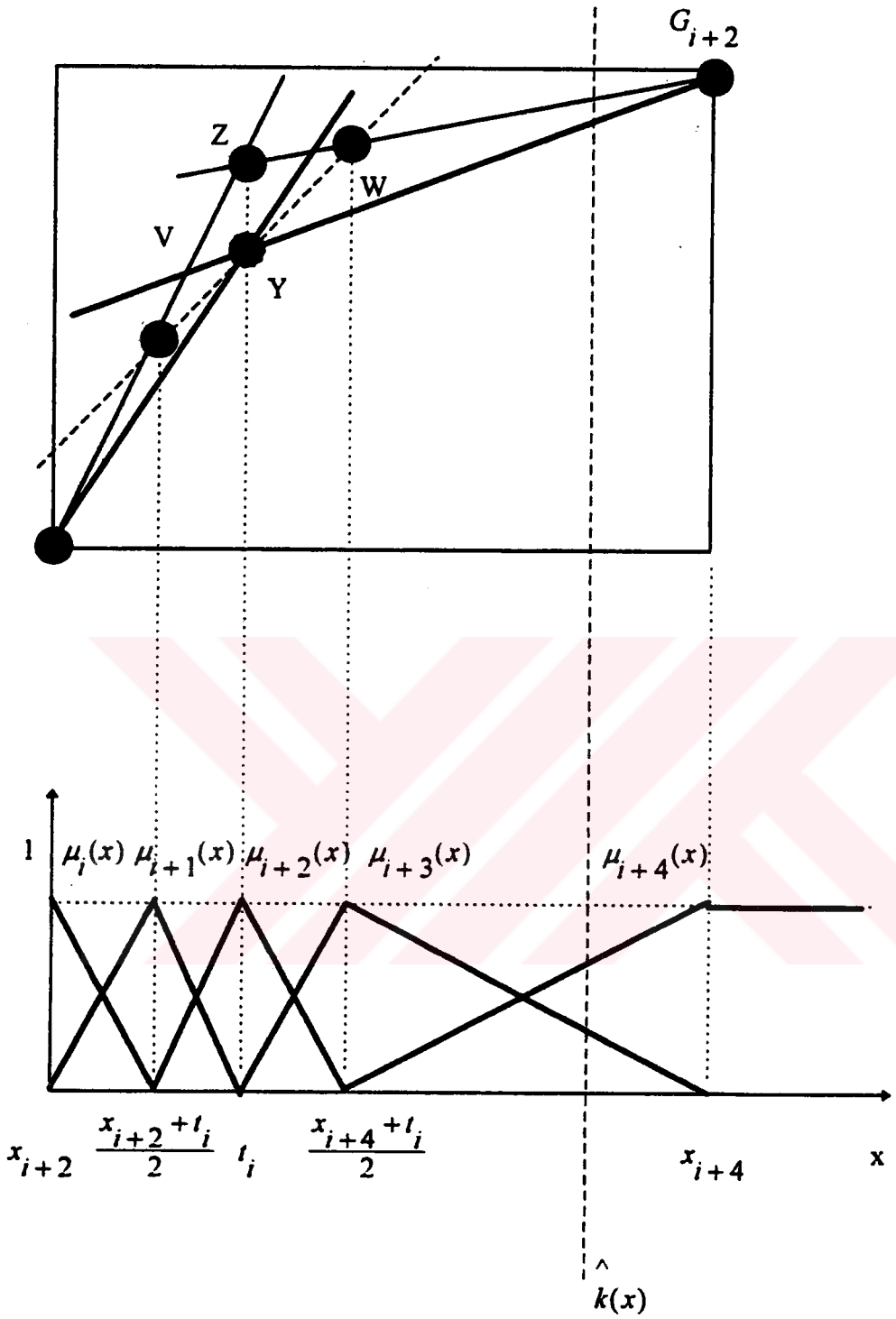
Kural 3: Eğer $t_i \le x < \frac{x_{i+4} + t_i}{2}$ ise,

$$\hat{k}(x) = \frac{f_{i+4}(x)\mu_{i+2}(x) + f_{i+1}(x)\mu_{i+3}(x)}{\mu_{i+2}(x) + \mu_{i+3}(x)} \tag{24}$$

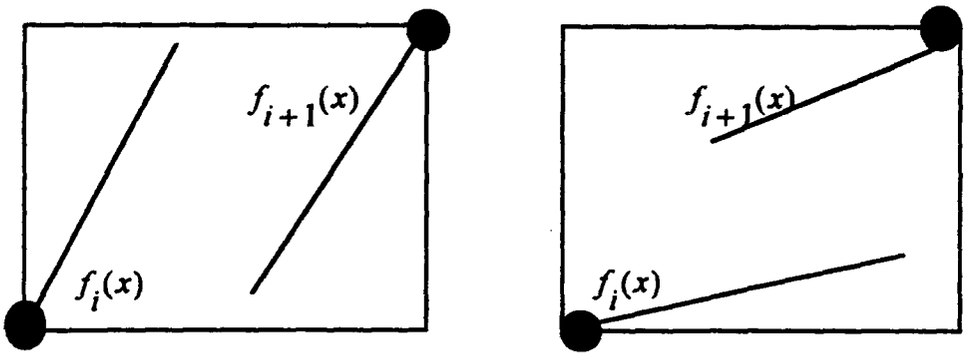
Kural 4: Eğer $\frac{x_{i+4} + t_i}{2} \le x \le x_{i+4}$ ise,

$$\hat{k}(x) = \frac{f_{i+2}(x)\mu_{i+3}(x) + f_{i+4}(x)\mu_{i+4}(x)}{\mu_{i+3}(x) + \mu_{i+4}(x)} \tag{25}$$

olur.



Şekil 22. $f_i(x)$, $f_{i+1}(x)$, $f_{i+2}(x)$, $f_{i+3}(x)$ ve $f_{i+4}(x)$ fonksiyonları ve $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları



Şekil 23. $f_i(x)$ ve $f_{i+1}(x)$ doğrularının D_{i+1} dikdörtgeninin içerisinde kesişmemesi durumları

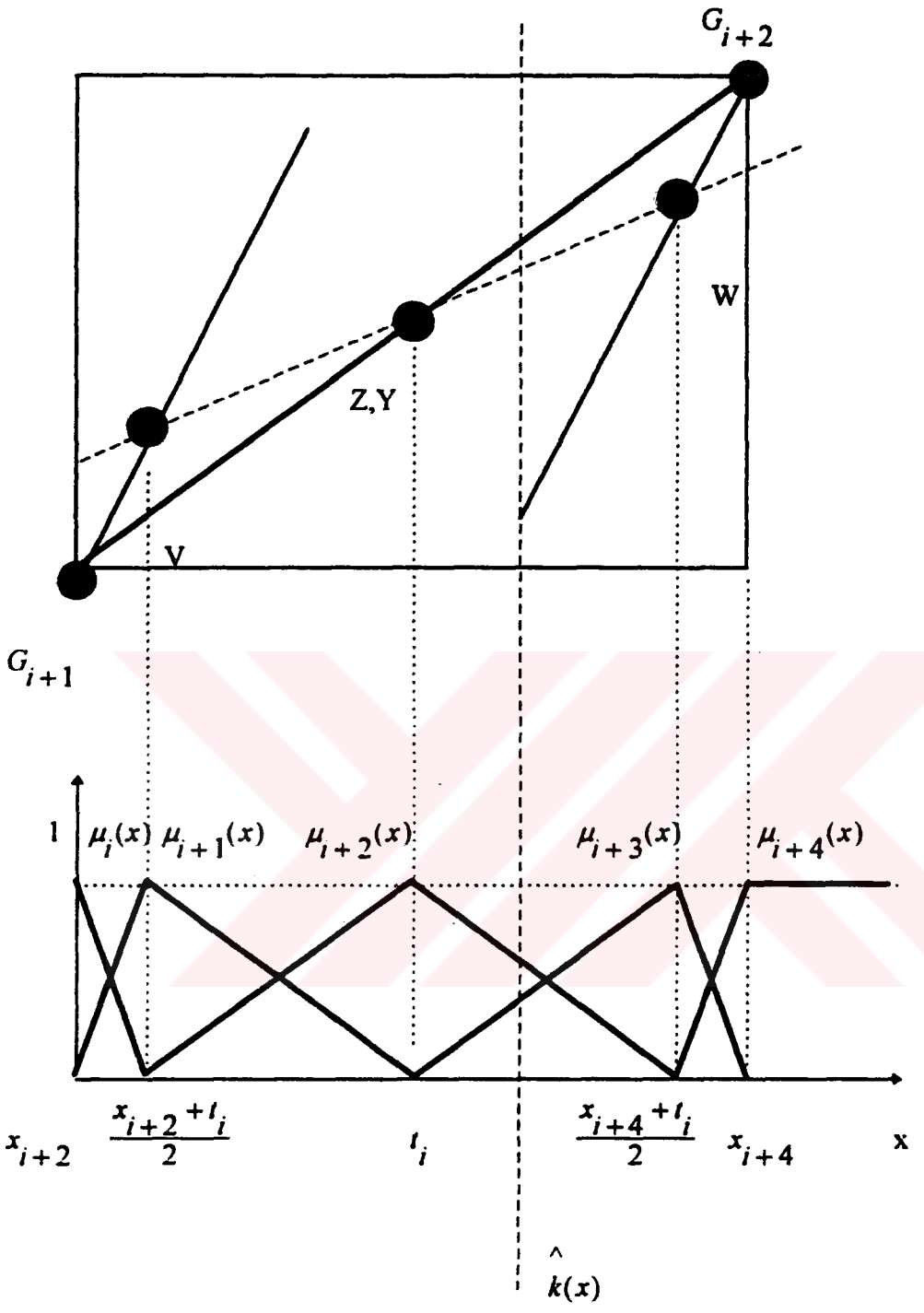
i) $f_i(x)$ ve $f_{i+1}(x)$ doğruları D_{i+1} dikdörtgeninin içerisinde kesişmesinler. (Şekil23) Bu durumda

$$t_i = \frac{x_{i+2} + x_{i+4}}{2} \tag{26}$$

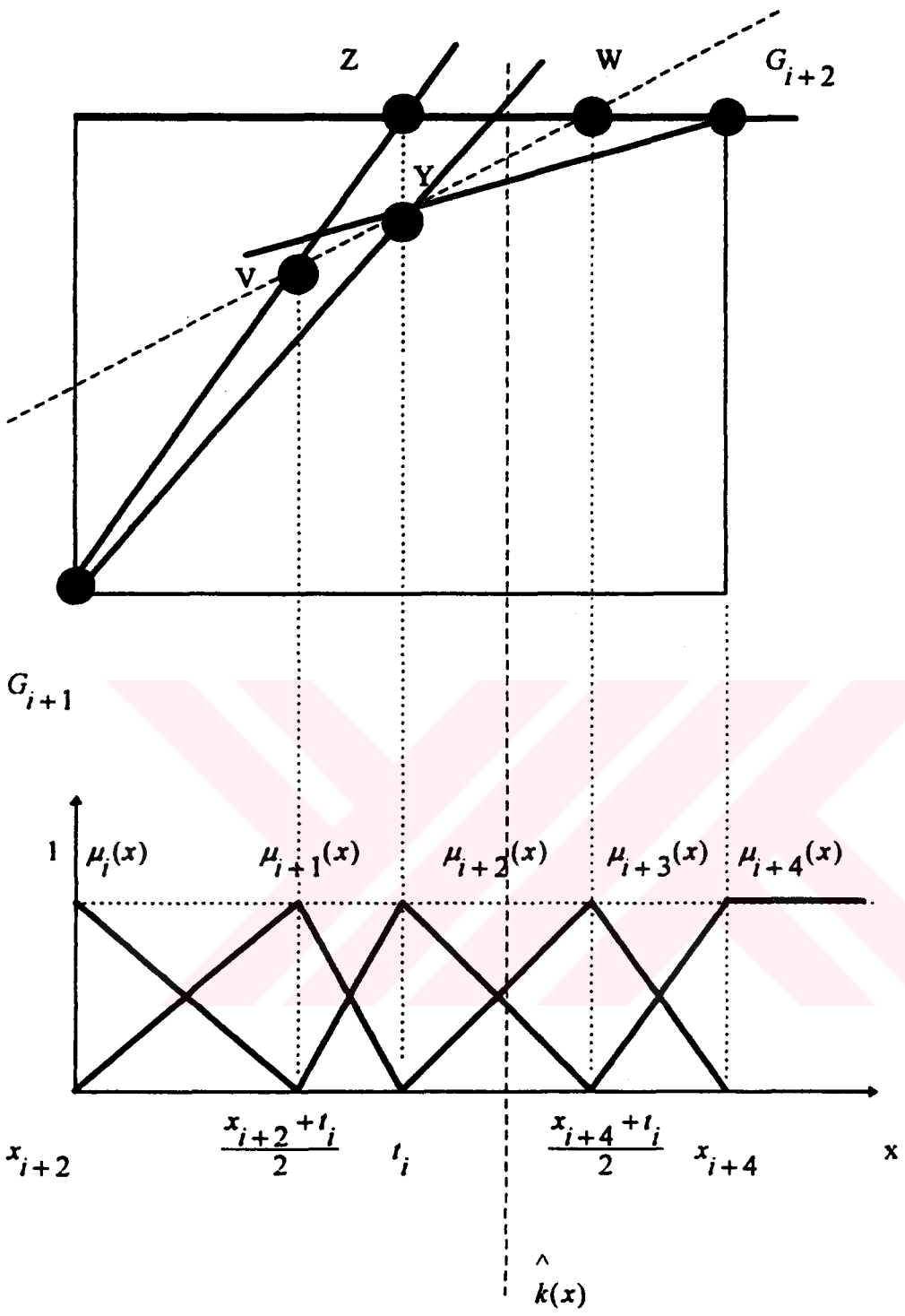
olarak alınır. Şekil 24'den de görüldüğü gibi Z ve Y koordinatları aynidir. Bu durumda ,$f_{i+3}(x)$ ve $f_{i+4}(x)$ doğruları da ayni doğru olur. $[x_{i+2}, x_{i+4}]$ aralığındaki aradeğerleme h) şıkkında olduğu gibi yapılır.

j) $m_i = 0$ olması durumunda, $f_i(x)$ doğrusu sıfır eğimle (x_{i+2}, g_{i+1}) noktasından, $m_{i+1} = 0$ olması durumunda ise, $f_{i+1}(x)$ doğrusu sıfır eğimle (x_{i+4}, g_{i+2}) noktasından geçer. Şekil 25'de yalnızca $m_{i+1} = 0$ durumda doğrular ve üyelik fonksiyonları gösterilmiştir.

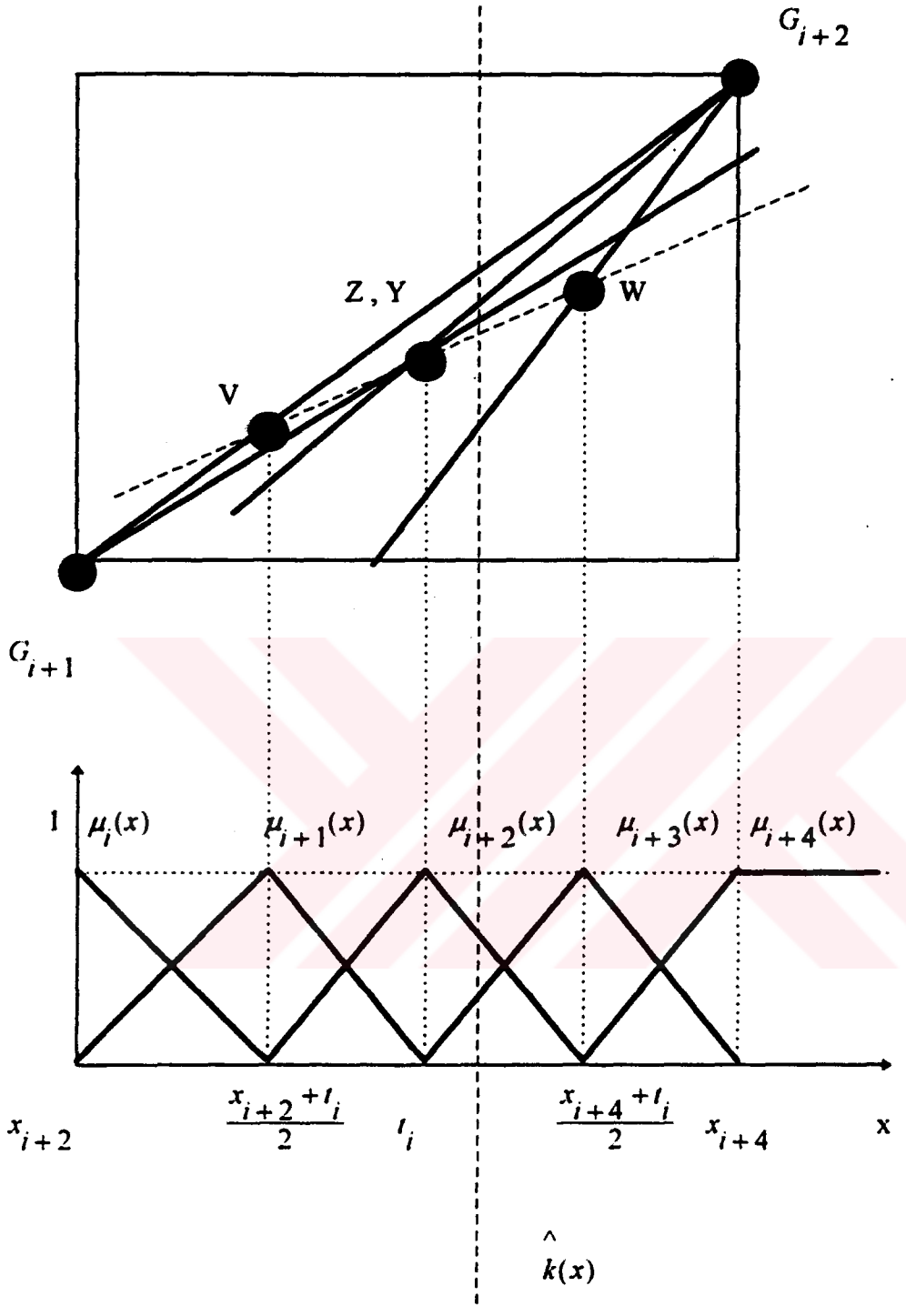
k) Eğer $s_i = s_{i+1} \neq 0$ ise $f_i(x)$ doğrusu s_i eğimi ile (x_{i+2}, g_{i+1}) noktasından geçer. $m_{i+1} \neq 0$ olduğu varsayılarak, bu durum için doğrular ve üyelik fonksiyonları şekil 26'da gösterilmiştir.



Şekil 24. $f_i(x)$, $f_{i+1}(x)$, $f_{i+2}(x)$, $f_{i+3}(x)$ ve $f_{i+4}(x)$ fonksiyonları ve $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları



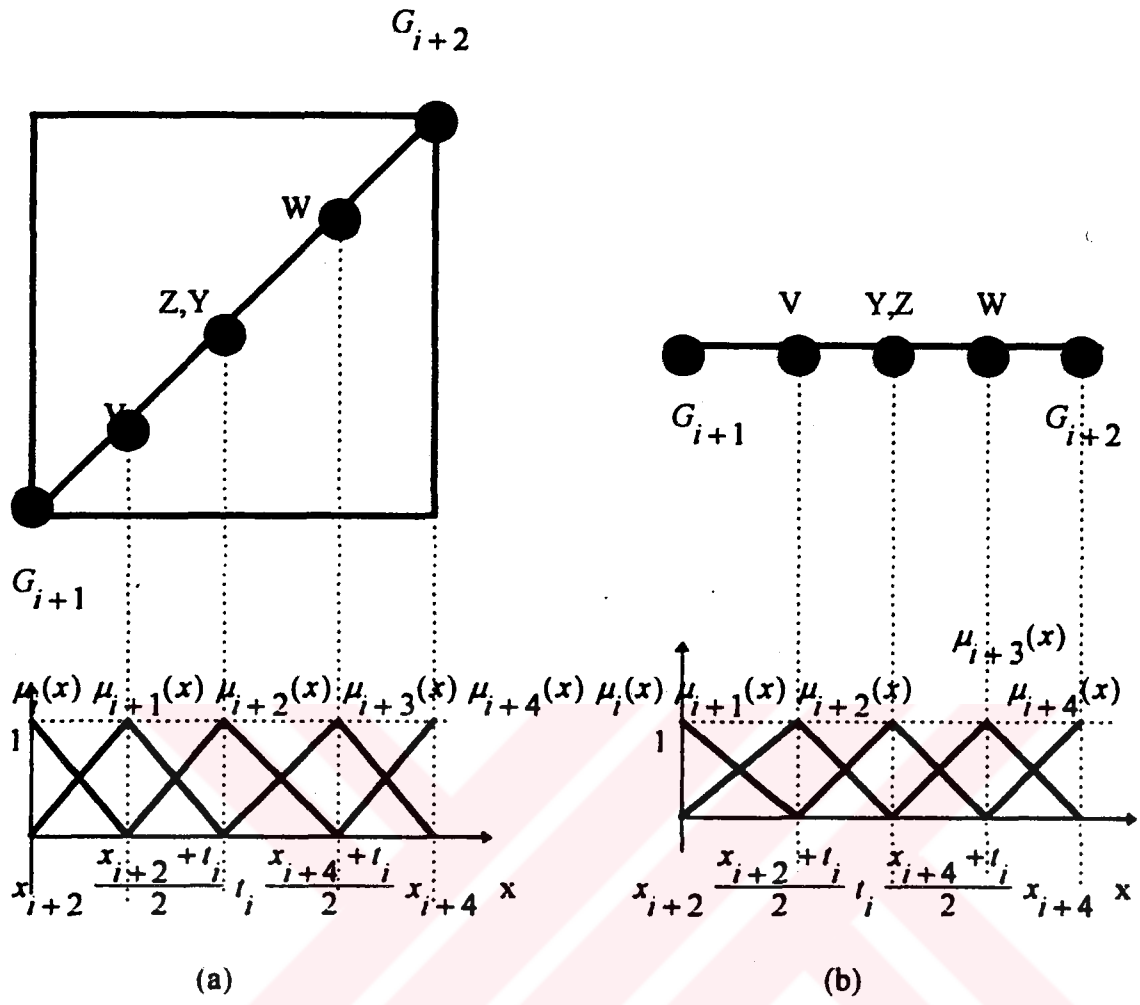
Şekil 25. $m_{i+1} = 0$ olması durumunda $f_i(x)$, $f_{i+1}(x)$, $f_{i+2}(x)$, $f_{i+3}(x)$ ve $f_{i+4}(x)$ fonksiyonları ve $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları



Şekil 26. $s_i = s_{i+1} \neq 0$ olması durumunda $f_i(x)$, $f_{i+1}(x)$, $f_{i+2}(x)$, $f_{i+3}(x)$ ve $f_{i+4}(x)$ fonksiyonları ve $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları

l) $f_i(x)$ doğrusu s_i eğimi ile (x_{i+2}, g_{i+1}) noktasından ve $f_{i+1}(x)$ doğrusu s_{i+1} eğimi (x_{i+4}, g_{i+2}) noktasından geçiyorsa, $f_i(x) = f_{i+1}(x) = f_{i+2}(x) = f_{i+3}(x) = f_{i+4}(x)$ olur. Bu durumda $t_i = \frac{x_{i+2} + x_{i+4}}{2}$ olarak alınır.

m) Hem $m_i = 0$ hem de $m_{i+1} = 0$ olması durumunda $f_i(x) = f_{i+1}(x) = f_{i+2}(x) = f_{i+3}(x) = f_{i+4}(x)$ doğruları sıfır eğimle (x_{i+2}, g_{i+1}) noktasından geçerler. Bu durumda $t_i = \frac{x_{i+2} + x_{i+4}}{2}$ olarak alınır. Şekil 27.b)'de bu durum gösterilmiştir.



Şekil 27. a) $f_i(x)$ doğrusu s_i eğimi ile (x_{i+2}, g_{i+1}) noktasından ve $f_{i+1}(x)$ doğrusu s_{i+1} eğimi (x_{i+4}, g_{i+2}) noktasından geçmesi durumunda $f_i(x) = f_{i+1}(x) = f_{i+2}(x) = f_{i+3}(x) = f_{i+4}(x)$ fonksiyonları ve $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları

b) Hem $m_i = 0$ hem de $m_{i+1} = 0$ olması durumunda $f_i(x) = f_{i+1}(x) = f_{i+2}(x) = f_{i+3}(x) = f_{i+4}(x)$ fonksiyonları ve $\mu_i(x)$, $\mu_{i+1}(x)$, $\mu_{i+2}(x)$, $\mu_{i+3}(x)$ ve $\mu_{i+4}(x)$ üyelik fonksiyonları

3. BULGULAR

Bu çalışmada, giriş fonksiyonuna eklenen çeşitli gürültü düzeyleri için çıkışta giriş fonksiyonunun gürültüsüz örnek değerlerinin elde edilmesine çalışılmıştır. Giriş fonksiyonundaki bazı örnek değerlerinin atılması; örneğin bir örnek değeri alınıp, bir örnek değeri alınmaması ya da bir örnek değeri alınıp, iki örnek değeri alınmaması gibi durumlar da incelenmiştir.

Geliştirilen bulanık mantık yöntemi, giriş işaretinin alçak geçiren bir süzgeçten geçirilerek çıkış işaretinin elde edilmesi işlemidir.

3.1. Performans Analizi ve Simülasyon Çalışması

Yapılan çalışmanın performans analizi için, "a" harfinin kayıt yapıldığı Win'95 altında çalışan wav uzantılı bir ses dosyasındaki örnek değerleri alınmıştır. Kaydedilen sesin örnekleme frekansı 11 KHz ve band genişliği de yaklaşık olarak 1.5 KHz'dir. Elde edilen işaretin ilk 1000 örnek değeri gözönünde bulundurulmuş ve bu değerlere göre işlem yapılmıştır. Bu 1000 örnek değeri 5 gruba ayrılmış, yani her 200 örnek değerine farklı gürültü seviyeli Gauss gürültüsü eklenme imkanı sağlanmıştır. Girişteki orijinal ses işaretinin standart sapması bulunmuş ve buna bağlı olarak eklenmek istenilen gürültü seviyesi işaret/gürültü oranından bulunarak ses işaretinin örnek değerlerine eklenmiştir.

Analiz için iki yöntem karşılaştırılmıştır. Geliştirilen yönteme önerilen yöntem, diğer yönteme ise Uchino yöntemi diyeceğiz.

İşaret / gürültü oranı, işaretin gücünün, işarete eklenen Gauss gürültüsünün değişintisine oranı olarak alındı. Elde edilen bazı sonuçlar , önerilen yöntem için çizelge 1'de, Uchino yöntemi için çizelge 2'de verilmiştir. İşarete istatistiği devamlı olarak değişen bir Gauss gürültüsü eklenmiş ve ortalama işaret / gürültü oranı çizelgelerde gösterilmiştir. Çizelgelerdeki bulunan değerler, aradeğerlenmiş işaret için işaret / gürültü oranını göstermektedir ve

$$İşaret / gürültü = 10\log_{10}[\sigma_i^2 / \sigma^2_{i-a}] \quad [dB] \qquad (27)$$

şeklinde hesaplanmıştır. Burada σ^2_i: orijinal işaretin değişintisi, σ^2_{i-a}: aradeğerlenen işaretle orijinal işaret arasındaki hatanın değişintisidir. Çizelgelerde verilen aradeğerlenen nokta sayısı, bulanık kural sayısı ve işarete eklenen ortalama işaret / gürültü oranlarına göre değerlenen işaretin işaret / gürültü oranları gösterilmiştir. 2N ve 3N ise, işaretin ayrıldığı alt parçalardaki örnek noktası sayısıdır. Önerilen yöntemin özellikle düşük işaret/gürültü oranlarında daha iyi sonuç verdiği görülmektedir. Tüm şekillerde gösterilen hata grafikleri, orijinal işaretteki örnek değerinin bulanık mantıkla bulunan örnek değerinden çıkartılıp mutlak değerinin alınması durumuna aittir. Şekil 28 - 30'da bu durumlar gösterilmiştir.

Süzgeçleme işlemi için girişe $x(t) = \sin(2\pi ft)$ işareti uygulanmıştır ve işaretin örnekleme frekansı $f_{ö}$ = 11 KHz alınmıştır. Çıkış işareti $y(t)$ gözlenmiştir. Bulanık süzgeçlerin frekans karakteristikleri süzgeçlerin kazançları gözönüne alınarak gösterilmiştir.

$$\text{Kazanç} \equiv 10\log_{10}(\langle y^2\rangle / \langle x^2\rangle) \qquad (28)$$

Burada y: bulanık mantık ile kestirilen çıkış işareti, x: gürültüsüz giriş işaretidir. Giriş işaretinin başlangıç frekansı 10 Hz alınmış ve her defasında 100 Hz artırılmıştır. Süzgeç karekteristikleri şekillerinden de görüleceği gibi; kesim frekansı, dikdörtgen içerisine alınan örnek nokta sayına bağlı olarak değişmektedir. Alınan örnek sayısı azaldıkça süzgeçin kesim frekansı da bu bağlı olarak artmaktadır. Bu durumlar şekil 31'de gösterilmiştir.

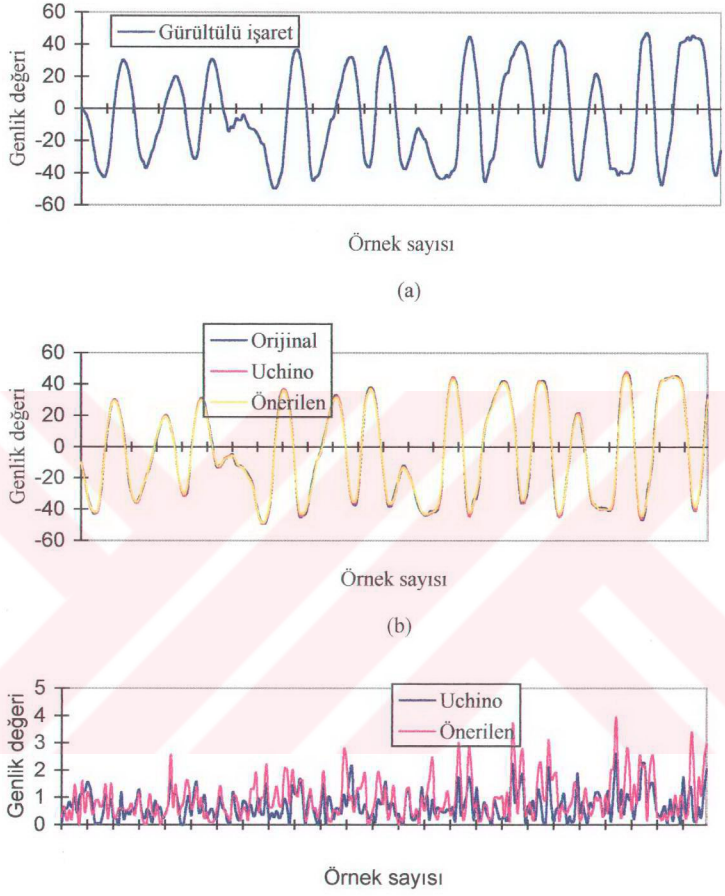
Giriş işaretine eklenen gürültü bir alçak geçiren süzgeçten geçirildikten sonra giriş işaretine eklenmesi durumunda, bulanık mantık süzgeçiyle çıkış işaretleri kestirilmeye çalışılmıştır. Bu durumlar şekil 32'de gösterilmiştir. Bütün şekillerde işaretin ilk 250 * aradeğerleme nokta sayısı örnek değeri gösterilmiştir. Yalnızca orijinal işaretin fourier dönüşümü ve süzgeçlenmiş gürültü işaretinin grafiğinde örnek değeri alınmıştır.

Çizelge 1. Önerilen yöntemle elde edilen bazı gürültü azaltımı sonuçları

Yöntem	Önerilen yöntem			
Aradeğerleme için nokta ve bulanık kural sayısı	1 nokta ve 5 kural		2 nokta ve 5 kural	
Dikdörtgendeki örnek sayısı	2N	3N	2N	3N
35 dB	35.64	27.17	25.95	16.82
30 dB	32.14	27.12	25.31	16.86
25 dB	28.31	25.95	24.37	16.80
20 dB	23.90	23.02	22.32	16.16
15 dB	19.46	20.81	19.10	15.96
10 dB	14.05	16.32	12.50	13.29
5 dB	9.34	10.35	8.97	9.11

Çizelge 2. Uchino yöntemiyle elde edilen bazı gürültü azaltımı sonuçları

Yöntem	Uchino yöntemi			
Aradeğerleme için nokta ve bulanık kural sayısı	1 nokta ve 5 kural		2 nokta ve 5 kural	
Dikdörtgendeki örnek sayısı	2N	3N	2N	3N
35 dB	37.95	31.40	29.73	20.40
30 dB	32.73	30.01	28.35	20.29
25 dB	27.84	27.41	26.03	19.83
20 dB	23.16	22.83	22.58	18.46
15 dB	18.49	20.25	18.85	17.23
10 dB	12.99	14.95	11.50	13.26
5 dB	8.06	8.79	7.80	8.22

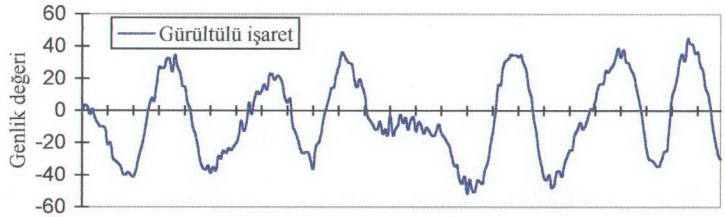


Şekil 28. 2N, 5 bulanık kural, 2 nokta aradeğerleme ve işaret / gürültü oranı 30 dB için giriş , çıkış ve hata işaretlerinin grafikleri

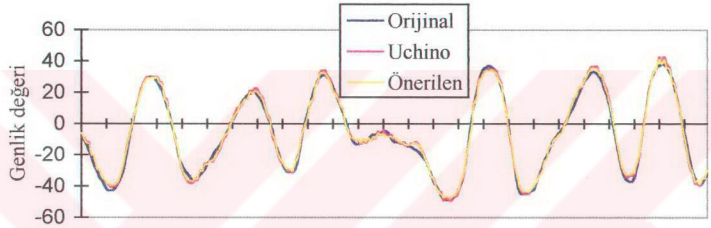
a) Gürültülü giriş işareti

b) Orijinal işaret, Uchino ve Önerilen yöntemle bulunan işaretler

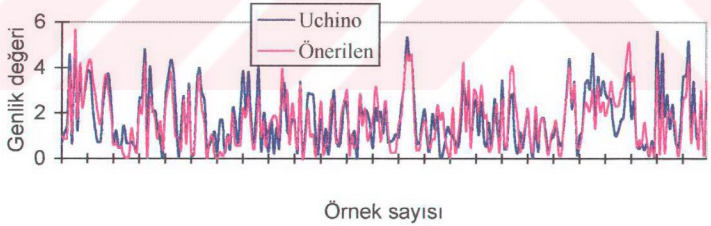
c) Her iki yöntem sonuçunda elde edilen hata işaretleri



(a)



(b)



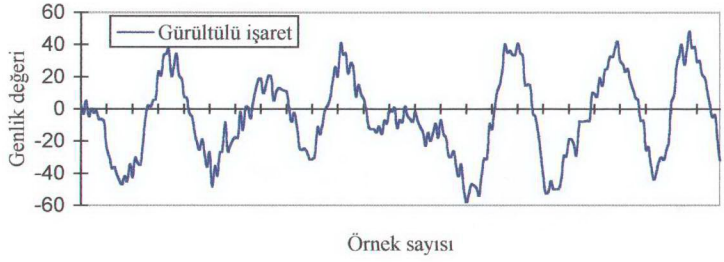
(c)

Şekil 29. 3N, 5 bulanık kural, 1 nokta aradeğerleme ve işaret / gürültü oranı 20 dB için giriş , çıkış ve hata işaretlerinin grafikleri

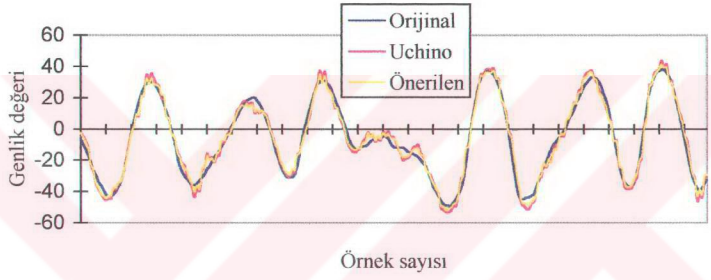
a) Gürültülü giriş işareti

b) Orijinal işaret, Uchino ve Önerilen yöntemle bulunan işaretler

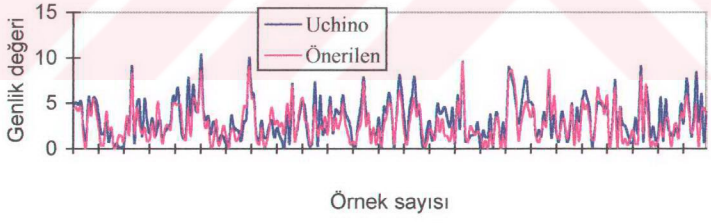
c) Her iki yöntem sonuçunda elde edilen hata işaretleri



(a)



(b)



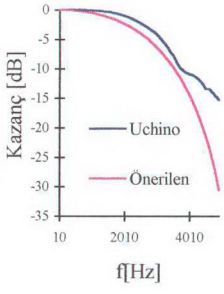
(c)

Şekil 30. 3N, 5 bulanık kural, 1 nokta aradeğerleme ve işaret / gürültü oranı 15 dB için giriş , çıkış ve hata işaretlerinin grafikleri

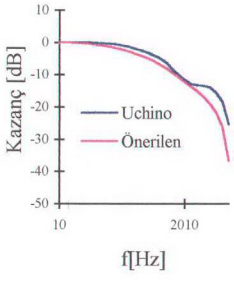
a) Gürültülü giriş işareti

b) Orijinal işaret, Uchino ve Önerilen yöntemle bulunan işaretler

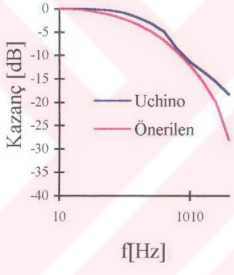
c) Her iki yöntem sonuçunda elde edilen hata işaretleri



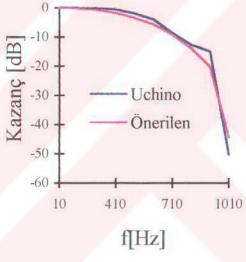
(a)



(b)



(c)



(d)

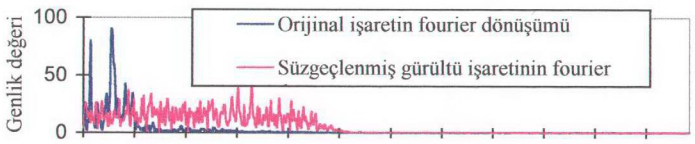
Şekil 31. $f_{\ddot{o}}$ = 11KHz'de süzgeçlerin karakteristikleri

a) 2N için

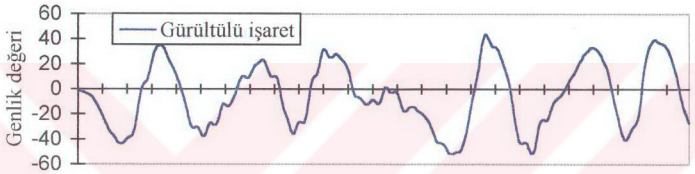
b) 3N için

c) 5N için

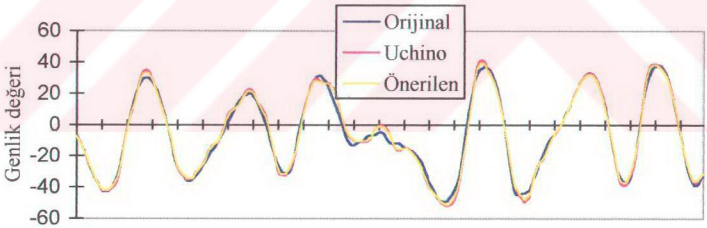
d) 7N için



(a)



(b)



(c)

Şekil 32. 3N, 5 bulanık kural, 1 nokta aradeğerleme ve işaret / gürültü oranı 15 dB için giriş , çıkış ve hata işaretlerinin grafikleri

a) Orijnal ve süzgeçlenmiş gürültü işaretinin fourier dönüşümü

b) Gürültülü giriş işareti

c) Orijinal işaret, Uchino ve Önerilen yöntemle bulunan işaretler

4. İRDELEME

Bu çalışmada, tek boyutlu bir gürültülü giriş işaretinin çıkışta gürültüsüz duruma yakın bir şekilde elde edilmesi için bir bulanık mantık yöntemi geliştirilmiştir. Geliştirilen yöntemin performans analizi aşağıdaki gibi irdelenebilir:

1) Girişe uygulanan işarete çeşitli değişken seviyelerde gürültü eklenerek çıkış işareti kestirilmeye çalışılmıştır. Bu yöntemlerde bulanık mantık kullanıldığı için giriş işaretin bütün örnek değerlerinin bilinmesine gerek yoktur. Örnek değerlerinin bir kısmı işleme tabi tutularak işaret kestirilmeye çalışılmıştır. Giriş işaretine eklenen toplamsal gürültünün istatistiğinin bilinmesine de gerek duyulmamaktadır.

2) Burada göz önüne alınan iki yöntem vardır. Uchino ve önerilen yöntemde, işleme tabi tutulmak için dikdörtgen içersine alınan örnek sayıları, değişken olabilmektedir. Önerilen yöntemde ele alınan bulanık mantık kural sayısı iki tane ve sabittir. Fakat Uchino yönteminde bu kural sayısı değişebilmektedir. İşleme tabi tutulan fazla kural ve örnek sayısı daha doğruluklu sonuçlar elde edilmesine sebep olarak gösterilebilir.

Her iki yöntemde de örnek sayısı, daha doğrusu dikdörtgen içerisine alınan nokta sayısı değiştirilerek ve giriş işaretine farklı seviyelerde gürültü eklenerek çıkış işaretinin ne kadar hata ile kestirileceği gözlenmiştir. Her iki yöntemde de nokta sayısı artırıldıkça işaretin kestirilmesindeki hata da artmaktadır. Uchino yönteminde işaretin yerel minimum ve yerel maksimum noktalarındaki kestirim daha az hata ile kestirilirken, önerilen yöntemde bu noktalardaki değerler biraz daha hatalı kestirilmektedir. Bunun nedeni ise, yeni yöntemde dikdörtgen içersine alınan noktaların ortalama değerlerinin alınarak işleme tabi tutulmasıdır. Fakat işarete eklenen gürültünün seviyesi artırıldığında yeni yöntem eski yönteme göre daha doğruluklu sonuç vermektedir.

Uchino yönteminde 5, 7, 9 bulanık kural kullanılarak sonuçlar bulunmuşken, yeni yöntem işleme tabi tutulan bulanık kural, 2 tane ve sabittir. Uchino yönteminde bulanık kural sayısı artırıldığında sonuçların daha iyi olması beklenirken, 5 bulanık kural kullanıldığında en iyi sonuç verdiği gözlenmiştir. Önerilen yöntemde kural sayısı sabit olduğundan işleme konan örnek değeri nokta sayısı*4 kadardır, örneğin nokta

sayısı 3 iken, 3*4 = 12 örnek değeri işleme konmuştur. Uchino yönteminde ise işleme konan örnek değeri en az nokta sayısı*6 kadardır, örneğin nokta sayısı 3 iken, 3*6 =18 örnek değeri işleme konmuştur.

Giriş işaretinden örnek değerler seçerek, örneğin bir örnek alıp bir örnek almama gibi, çıkışta işaret kestirilmeye çalışılmıştır ve çıkış işareti gözlenmiştir. Örnek aralığı ne kadar fazla seyrekleştirilirse, işareti kestirme her iki yöntemde de o kadar bozuk olduğu görülmüştür.

3) Her iki yöntemin nasıl bir süzgeç karakteristiği gösterdiği gözlenmek için girişe frekansı değiştirilebilen bir sinüs uygulanmış ve çıkıştaki işaret gözlenmiştir. Her iki yöntem içinde bulanık mantık süzgeçlerinin kazançları bulunmuş ve bu değerlere bağlı olarak bu süzgeçlerin birer alçak geçiren süzgeç karakteristiği gösterdiği görülmüştür.

5. SONUÇLAR

Bu çalışmada, gürültüsüz ve toplamsal gürültü içeren örneklenmiş işaretler için bulanık mantığa dayalı bir aradeğerleme algoritması geliştirildi. Bu yöntemle, çevre noktalardaki örnek değerleri bulanık mantığa dayalı algoritmada kullanılarak bulunmak istenen noktadaki örnek değeri kestirildi. Gürültülü işaretler için aradeğerleme yapılırken, gürültü içeren komşu örnek noktaların bir dikdörtgensel bölge içersine alınıp bir sınıflandırılması yapıldı. Bu yöntem, değişik örnekleme hızlarında ve gürültü seviyelerinde bilgisayar simülasyonu ile incelendi. Yapılan çalışmalar aşağıdaki sonuçlara varılmıştır:

1) Bulanık mantık yöntemi kullanılarak, istatistiği bilinmeyen gürültülü veya gürültüsüz örneklenmiş işaretlerin kestirilmesi işlemi daha az zaman harçanarak yapılabilir.

2) İşaret kestirilmeye çalışılırken giriş işaretin bütün örnek değerlerinin bilinmesine gerek duyulmamaktadır. İşaretin kullanılan metoda göre bazı örnek değerlerine göre işlem yapılmaktadır.

3) Uchino ve önerilen yöntemin aslında birer doğrusal olmayan alçak geçiren süzgeç karekteristiği gösterdiği saptanmıştır.

4) Her iki yöntemin de basit oluşu, sistemin matematiksel modellerinin bilinmesine gerek duyulmaması ve elde edilen sonuçların da diğer yöntemlere göre daha iyi olması, bulanık mantığın diğer alanlarda olduğu gibi işaret örneklemede de kullanılabileceğini göstermektedir.

6. KAYNAKLAR

1. Ohta, M., Uchino, E., A Design for a General Digital Filter For State Estimation of an Arbitrary Stochastic Sound System, J. Acoust. Soc. Of America, 80 (1996) 804-812.

2. Kalman, R., E., Bucy, R., S., New Results in Linear Filtering and Prediction Theory, Trans. ASME, Ser. D. J. Basic Engineering, 83 (1961) 95-108.

3. Uchino, E., Yamakawa, T., Miki, T. ve Nakamura, S., "Fuzzy Rule-Based Simple Interpolation Algorithm For Discrete Signal", Fuzzy Sets and Systems, 59 (1993) 259-270.

4. Dizdaroğlu, B., Gangal, A., Kayıkçıoğlu, T. ve Atasoy, A., İşaret Örneklemede Bulanık Mantık Yöntemi, 5. Sinyal İşleme ve Uygulamaları Kurultayı, 1-3 Mayıs 1997, Kuşadası, Bildiriler Kitabı, Cilt II, 855 - 860.

5. Chen, J.E. ve Otto, K.E., "Constructing Membership Functions Using Interpolation, and Measurement Theory", Fuzzy Sets and Systems, 73 (1995) 313-327.

6. Günal, Ü.,"Bulanık Mantık", Otomasyon, 55 (1997) 50-55.

7. Ross, T.J., Fuzzy Logic With Engneering Applications, McGraw-Hill, Inc, USA, 1995.

8. Bandemer, H. ve Gottwald, S., Fuzzy Sets, Fuzzy Logic, and Fuzzy Methods With Applications, John Wiley & Sons Ltd, West Sussex, 1995.

9. Klir, G.J, Folder, T.A., Fuzzy Sets, Uncertainty, and Information, Prentice Hall, New Jersey, 1988.

7. EKLER

Ek 1. Bilgisayar programının C ++ kaynak kodu dosyası

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include <math.h>
#include <windows.h>
#include <iostream.h>
#include <commdlg.h>
#include <mem.h>

#pragma hdrstop
#include "fuzzy.h"
#include "bitirme.rh"
#define boyut 1000

typedef double eleman[boyut];
eleman dizi,gec_dizi,h;
double cik_deg[2000];

int noktasayi,kac_mem_ship,fzy_kural,artim,secimOK,bul_deg;
double gur_db[5],yen_hata,esk_hata,ciz_deg;
char yen_ileti[20],esk_ileti[20],kal_ileti[20],gec_dur[20];

char top_ileti[50],yen_ile_top[20],esk_ile_top[20];

char dos_ism[80];
typedef double deger[15];
void suzgecleme();
```

```
void std_sap_oku(const char *,double &);
BOOL CALLBACK DialogG(HWND,UINT, WPARAM, LPARAM);
BOOL CALLBACK DialogY(HWND,UINT, WPARAM, LPARAM);

LRESULT CALLBACK WindowFunc(HWND,UINT, WPARAM,LPARAM);

char szWinName[]="MyWin";
int maxX,maxY;
long status1=0,status2=0,status3=0,status4=0,status5=0,status6=0,status7=0,
        status8=0;

HDC memdc;
HBITMAP hbit;
HBRUSH hbrush,hOldbrush;
HPEN hOldpen,hRedpen,hBluepen,hGreenpen,hGrypen,hYellowpen,hWhitepen,
        hOrangepen,hBrownpen;
HINSTANCE hInst;

//dosya text modunda açılıp dosyadan istenilen miktarda okuma yapılıyor.
void dos_text_foroku_ac(const char *dosyaadi, int dizi_bas_elm,
                        int dizi_son_elm,int artim,eleman dizi)
{
        int i,j;
        FILE *dosya;
        double ara_deg;

        flushall();
        if((dosya = fopen(dosyaadi,"r"))==NULL)
         exit(1);
```

```
        fscanf(dosya,"%lf\n",&dizi[dizi_bas_elm]);
//      printf("%lf\n",dizi[dizi_bas_elm]);
        for(i=dizi_bas_elm+1;i<dizi_son_elm;i++)
         {
         for(j=0;j<artim;j++)
                fscanf(dosya,"%lf\n",&ara_deg);
         dizi[i]=ara_deg;
        // printf("%d = %lf\n",i,dizi[i]);
         }//for
        fclose(dosya);
}
void dos_text_foroku(const char *dosyaadi)
{
        int i;
        FILE *dosya,*dosya1;
        double ara_deg;

        flushall();
        if((dosya = fopen(dosyaadi,"r"))==NULL)
         exit(1);

        dosya1 = fopen("c:\\user\\bekir\\deg1.dat","w");

        for(i=0;i<2000;i++)
         {
                fscanf(dosya,"%lf\n",&ara_deg);
                fscanf(dosya,"%lf\n",&ara_deg);
                fprintf(dosya1,"%lf\n",ara_deg);
         }//for
        fcloseall();
}
```

```
//dosya text modunda açılıp, dosyaya istenilen miktarda yazma yapılıyor.
 void dos_text_foryaz_ac(const char *dosyaadi,int dizi_bas_elm,
                         int dizi_son_elm,int artim)
{
        int i;
        FILE *dosya;

        flushall();
        if((dosya = fopen(dosyaadi,"w"))==NULL)
         exit(1);

        for(i=dizi_bas_elm;i<dizi_son_elm;i+=artim)
         fprintf(dosya,"%lf\n",dizi[i]);
        fclose(dosya);
}
//dosyaya  ekleme işlemi yapılıyor.
void dos_text_forekle(const char *dosyaadi1,const char *dosyaadi2,
              const char *dosyaadi3,int dizi_bas_elm,int dizi_son_elm,int deger)
{
        int i;
        FILE *dosya1,*dosya2,*dosya3;
        double aradegr1,aradegr2,aradegr;

        if((dosya1 = fopen(dosyaadi1,"r"))==NULL)
         exit(1);
        if((dosya2 = fopen(dosyaadi2,"r"))==NULL)
         exit(1);

        if(deger==0){
         if((dosya3 = fopen(dosyaadi3,"w"))==NULL)
         exit(1);}
```

```
        else {
                if((dosya3 = fopen(dosyaadi3,"a"))==NULL)
         exit(1);}

        flushall();
        for(i=0;i<dizi_son_elm;i++)
         {
         fscanf(dosya1,"%lf\n",&aradegr1);
         if(i>=dizi_bas_elm){
                fscanf(dosya2,"%lf\n",&aradegr2);
                aradegr=aradegr1+aradegr2;
                fprintf(dosya3,"%lf\n",aradegr);
         }//if
         }
        fcloseall();
}

//dikdörtgende max ve min degerler bulunuyor.
void dikdort_ic_nok(int nokta,int dizi_elm,double &max,double &min)
{
 int i;

        max = dizi[dizi_elm];
        min = dizi[dizi_elm];

        for(i=1;i<nokta;i++)
         if( dizi[dizi_elm + i] > max ) max = dizi[dizi_elm + i];
                else if( dizi[dizi_elm + i] < min ) min = dizi[dizi_elm + i];
        }
```

```
//g degeri bulunuyor.
void g_deger_bul(double d, double c,double &g)
{ g =(d+c)/2.0; }

//verilen iki nokta arasındaki deger bulunuyor
void fonk_deg_bul_egim(double m,double x,double t,double y,double &son_deg)
{ son_deg = m*(t-x)+y;}

void fonk_deg_bul_nokta(double y[],double x[],double t,double &son_deg)
{ son_deg=( (t-x[0])*(y[1]-y[0])+y[0]*(x[1]-x[0]) )/(x[1]-x[0]);}

//old fuzzy için ağırlık merkezine göre fonksiyon degerlerinin belirlenmesi.
void  old_agl_mer_gor_deg_bul(double d,double c,double b,double a,
                                                    double  t,double  ti,double
g[],double &sonuc)
{
//g[0]=gi-1,g[1]=gi+1
if((g[1]>g[0])&&(g[1]<g[2]))
sonuc=((d-c)*(t-ti)/(b-a))+g[1];

else if((g[1]<g[0])&&(g[1]>g[2]))
sonuc=-((d-c)*(t-ti)/(b-a))+g[1];

else sonuc=g[1];
}
//new fuzzy ici si degerlerinin bulunması y=d,c; x=a,b;
void  new_si_deg_bul(double d[],double c[],double b[],double a[],
                        int &sonuc,double si[])
{

int i;
```

```
        for(i=0;i<2;i++)
          {     si[i]=(d[i]-c[i])/(b[i]-a[i]);
          //    printf("%lf",si[i]);
          }
        if(si[0]*si[1] <= 0)
          sonuc=0;
        else if(fabs(si[0]) > fabs(si[1]))
          sonuc=1;
        else if(fabs(si[0]) < fabs(si[1]))
          sonuc=2;
        else if( si[0] == si[1])
          sonuc=3;
 //  printf("%d",sonuc);
 }

//fuzzy üyelik fonksiyonlarının belirlenerek sonucun bulunması.

void fzy_uyelik_fonk(double x[],double bul_x,double &cikis)
{
 auto triangular_membership MS(x[0],x[1],x[2]);

 auto fzy_set Uy_x_deg(1);
 auto double  Uy_y_deg;

 Uy_x_deg[0]=MS.membership(bul_x);
 Uy_y_deg=MS.weight(Uy_x_deg[0]);
 cikis=Uy_y_deg;
 // printf("Cik=%lf\n",cikis);

 }
```

```
//dikdörtgenlere karşılık gelen dogrulardaki degerler bulunup,dog_dege
//atanıyor.Kural sayısı gözönüne aliniyor. Kaç nokta olacağı ve başlangıçı ve
//bulunacak nokta gözönüne alınıyor.

void old_fzy_dogru_snc(int fzy_kural,double bul_x,
                       int noktasayi,int basl,deger dog_deg,double x_nokta[])
{
 double max,min,ga[3],g,sonuc;
 deger gd,ad,bd,cd,dd,td,tid;
 double a,b,c,d,t,ti,y_gecici[2],x_gecici[2];
 int i;

for(i=0;i<fzy_kural;i++)
        {
        dikdort_ic_nok(noktasayi,basl,max,min);
        g_deger_bul(max,min,g);

        gd[i]=g;
        dd[i]=max; cd[i]=min;
        bd[i]=basl+noktasayi-1; ad[i]=basl;

        td[i]=bul_x; tid[i]=(ad[i]+bd[i])/2.0;
        basl+=noktasayi-1;
        x_nokta[i]=tid[i];
              // printf("%lf",x_nokta[i]);
       }

for(i=1;i<fzy_kural-1;i++)
        {
        d=dd[i];
        c=cd[i];
```

```
        b=bd[i];
        a=ad[i];
        t=td[i];
        ti=tid[i];
        ga[0]=gd[i-1];
        ga[1]=gd[i];
        ga[2]=gd[i+1];
        old_agl_mer_gor_deg_bul(d,c,b,a,t,ti,ga,sonuc);
        dog_deg[i-1]=sonuc;
 }

y_gecici[0]=gd[fzy_kural/2-1];
y_gecici[1]=gd[fzy_kural/2];

x_gecici[0]=tid[fzy_kural/2-1];
x_gecici[1]=tid[fzy_kural/2];

fonk_deg_bul_nokta(y_gecici,x_gecici,bul_x,sonuc);
dog_deg[fzy_kural-2]=sonuc;
}

//Kesişip kesişmeyen dogruların bulunması.
void new_fzy_dog_kes(double si[][2],double mi[],int &sonuc)
 {
 if((fabs(si[0][1]) > fabs(mi[0])) && (fabs(si[0][1]) > fabs(mi[1])))
        sonuc=0;
 else if((fabs(si[0][1]) < fabs(mi[0])) && (fabs(si[0][1]) < fabs(mi[1])))
        sonuc=0;
 else sonuc=1;
}
```

```
//yeni fuzzy için gd[i] karsilastiriliyor.
void gd_karsilastir(double gd[],int &sonuc)
 {
        if((gd[6]>gd[5])&&(gd[6]<gd[7]))
                sonuc=0;
        else if((gd[6]<gd[5])&&(gd[6]>gd[7]))
                sonuc=1;
        else sonuc=2;
 }

//yeni fuzzy sonuçları.
void new_fzy_dogru_snc(double bul_x,
                int noktasayi,int basl,deger dog_deg,double x_nokta[])
 {
 double T,max,min,g,d[2],c[2],b[2],a[2],si[2],mi[2],bx,cx,gecici_si[2][2];
 deger gd,ad,bd,tid;
 int i,j,secim,gd_son;
 double y,x,t,m,y_gecici[2],x_gecici[2],sonuc;


 for(i=0;i<8;i++)
        {
         dikdort_ic_nok(noktasayi,basl,max,min);
         g_deger_bul(max,min,g);
         gd[i]=g;
         bd[i]=basl+noktasayi-1; ad[i]=basl;
         tid[i]=(ad[i]+bd[i])/2.0;
         basl+=noktasayi-1;
//       printf("%lf ",gd[i]);
 }
```

```
for(i=0;i<2;i++)
 {
        c[0]=gd[0+i];c[1]=gd[1+i];
        d[0]=gd[1+i];d[1]=gd[2+i];

        a[0]=tid[0+i];a[1]=tid[1+i];
        b[0]=tid[1+i];b[1]=tid[2+i];

        new_si_deg_bul(d,c,b,a,secim,si);

//si gecici değişkene atılıyor.
 for(j=0;j<2;j++)
        gecici_si[i][j]=si[j];

//mi,mi+1 bulunuyor.
 switch(secim){
        case 0:mi[i]=0.0;
                        // printf("%lf",mi[i]);
         break;

        case 1:bx=(gd[2+i]-gd[1+i])/si[0] + tid[1+i];
                        cx=(bx+tid[2+i])/2.0;
                        mi[i]=(gd[2+i]-gd[1+i])/(cx-tid[1+i]);
                       // printf("%lf",mi[i]);
         break;

        case 2:bx=(gd[0+i]-gd[1+i])/si[0] + tid[0+i];
                        cx=(bx+tid[0+i])/2.0;
                        mi[i]=(gd[1+i]-gd[0+i])/(tid[1+i]-cx);
                        //printf("%lf",mi[i]);
         break;
```

```
        case 3:mi[i]=si[0];
                        //printf("%lf",mi[i]);
         break;

        }//switch
// printf("%lf",mi[i]);
 }//for
//doğrular bulunacak.
//t degerleri bulunacak.
new_fzy_dog_kes(gecici_si,mi,secim);

 switch(secim) {
 //dogrular kesişmiyor.
 case 0:
                T=(tid[1]+tid[2])/2.0;
        // printf("%lf ",T);
        break;
 //dogrular kesişiyor.
 case 1:
                if(mi[0]!= mi[1])
                 T=(gd[2]-gd[1] + mi[0]*tid[1]-mi[1]*tid[2])/(mi[0]-mi[1]);
                if((T <= tid[1])||(T >= tid[2]))
                 T=(tid[1]+tid[2])/2.0;
        break;
 }//switch
// T=tid[2]+tid[2])/2.0;

// printf("%lf\n",T);

 x_nokta[0]=tid[1];
```

```
 x_nokta[1]=(T+tid[1])/2.0;
 x_nokta[2]=T;
 x_nokta[3]=(T+tid[2])/2.0;
 x_nokta[4]=tid[2];

/* printf("x0=%lf",x_nokta[0]);
 printf("x1=%lf",x_nokta[1]);
 printf("x2=%lf",x_nokta[2]);
 printf("x3=%lf",x_nokta[3]);
 printf("x4=%lf",x_nokta[4]);
 */
 //fi doğrusu için deger bulundu.
 y=gd[1];
 x=tid[1];
 m=mi[0];
 t=bul_x;
 fonk_deg_bul_egim(m,x,t,y,sonuc);
 dog_deg[0]=sonuc;
// printf("g1=%lf",gd[1]);
// printf("g2=%lf",gd[2]);
// printf("d=%lf\n",dog_deg[0]);

 //fi+1 dosrusu icin deger bulundu.
 y=gd[2];
 x=tid[2];
 m=mi[1];
 t=bul_x;
 fonk_deg_bul_egim(m,x,t,y,sonuc);
 dog_deg[1]=sonuc;
// printf("g1=%lf",gd[1]);
// printf("g2=%lf",gd[2]);
```

```
// printf("d=%lf\n",dog_deg[1]);

 //v noktası icin y degeri bulundu.
 y=gd[1];
 x=tid[1];
 m=mi[0];
 t=x_nokta[1];
 fonk_deg_bul_egim(m,x,t,y,sonuc);
 y_gecici[0]=sonuc;

// printf("g1=%lf",gd[1]);
// printf("g2=%lf",gd[2]);
// printf("d=%lf\n",sonuc);

 //w noktası icin y degeri bulundu.
 y=gd[2];
 x=tid[2];
 m=mi[1];
 t=x_nokta[3];
 fonk_deg_bul_egim(m,x,t,y,sonuc);
 y_gecici[1]=sonuc;

// printf("g1=%lf",gd[1]);
// printf("g2=%lf",gd[2]);
// printf("d=%lf\n",sonuc);

//fi+2 doğrusu için deger bulundu.
 x_gecici[0]=x_nokta[1];
 x_gecici[1]=x_nokta[3];
 t=bul_x;
 fonk_deg_bul_nokta(y_gecici,x_gecici,t,sonuc);
```

```
dog_deg[2]=sonuc;

// printf("g1=%lf",gd[1]);
// printf("g2=%lf",gd[2]);
// printf("d=%lf\n",dog_deg[2]);

//y noktasi bulunuyor.
t=x_nokta[2];
fonk_deg_bul_nokta(y_gecici,x_gecici,t,sonuc);
y=sonuc;
//printf("g1=%lf",gd[1]);
//printf("g2=%lf",gd[2]);
//printf("d=%lf\n",y);

//fi+3 doğrusu için değer bulunuyor.
y_gecici[0]=gd[1];
y_gecici[1]=y;
x_gecici[0]=tid[1];
x_gecici[1]=x_nokta[2];
t=bul_x;
fonk_deg_bul_nokta(y_gecici,x_gecici,t,sonuc);
dog_deg[3]=sonuc;
//printf("g1=%lf",gd[1]);
//printf("g2=%lf",gd[2]);
//printf("d=%lf\n",dog_deg[3]);


//fi+4 doğrusu için değer bulunuyor.
y_gecici[0]=gd[2];
y_gecici[1]=y;
x_gecici[0]=tid[2];
```

```
x_gecici[1]=x_nokta[2];
t=bul_x;
fonk_deg_bul_nokta(y_gecici,x_gecici,t,sonuc);
dog_deg[4]=sonuc;

//printf("g1=%lf",gd[1]);
// printf("g2=%lf",gd[2]);
// printf("d=%lf\n",dog_deg[0]);
}

//xg,xo,xs ile verilen uyelik fonk. tabanlari kullanılarak,bul_x degeri
//uyelik fonksiyonlarından bulunuyor.
void fzy_mem_ship_snc(int kac_mem_ship,double xg[],double xo[],

                double xs[],double bul_x,

                double fzy_uy_deg[])
{
 int i;
 double x[3],cikis;

 for(i=0;i<kac_mem_ship;i++)
        {
         x[0]=xg[i];
         x[1]=xo[i];
         x[2]=xs[i];

         fzy_uyelik_fonk(x,bul_x,cikis);
         fzy_uy_deg[i]=cikis;

        // printf("g=%lf\n",xg[i]);
```

```
        // printf("o=%lf\n",xo[i]);
        // printf("s=%lf\n",xs[i]);
        // printf("%lf\n",fzy_uy_deg[i]);

}
        // printf("x=%lf\n",bul_x);

}

void old_fzy_sonuc(const char *gd_adi,int ilk_deg,int son_deg,
         int noktasayi,int kac_mem_ship,int fzy_kural,int transfer_mi)
{
int i,d_nokta,j;
double
bul_x,basl,x_nokta[10],xg[10],xo[10],xs[10],fzy_uy_deg[10],fzy_bul,esk_h_top;

double fzy_pay,fzy_payda,is_std;
deger dog_deg;
FILE *dosya;
FILE *hata_dos;

esk_hata=0.0;esk_h_top=0.0;

if(transfer_mi)
{
 if((hata_dos=fopen("c:\\user\\bekir\\eskih.dat","w"))==NULL) exit(1);
 dos_text_foroku_ac(dos_ism,0,boyut,artim,gec_dizi);
}

if((dosya=fopen(gd_adi,"w"))==NULL) exit(1);
```

```
//gözönünde bulunacak degerler.
switch(fzy_kural){
 case 6:d_nokta=0;break;
 case 8:d_nokta=1;break;
 case 10:d_nokta=2;break;
 case 12:d_nokta=3;break;
 case 14:d_nokta=4;break;
 }

//i sinirlari belirlenecek.
flushall();
//cizim için başlangiç belirle
ciz_deg=ilk_deg;
for(i=ilk_deg;i<son_deg;i++){
 fzy_pay=0.0;
 fzy_payda=0.0;
 basl=i;
 bul_x=(3.0+d_nokta)*(noktasayi-1)+basl;
 old_fzy_dogru_snc(fzy_kural,bul_x,noktasayi,basl,dog_deg,x_nokta);

 xg[0]=basl;
 xo[0]=x_nokta[1];
 xs[0]=x_nokta[3];

 xg[1]=x_nokta[0];
 xo[1]=x_nokta[2];
 xs[1]=x_nokta[4];

 xg[2]=x_nokta[1];
 xo[2]=x_nokta[3];
 xs[2]=x_nokta[5];
```

```
xg[3]=x_nokta[2];
xo[3]=x_nokta[4];
xs[3]=x_nokta[5]+(noktasayi-1)/2;

fzy_mem_ship_snc(kac_mem_ship,xg,xo,xs,bul_x,fzy_uy_deg);

for(j=0;j<kac_mem_ship;j++)
{
        fzy_pay+=dog_deg[j]*fzy_uy_deg[j];
        fzy_payda+=fzy_uy_deg[j];
}
fzy_pay+=dog_deg[kac_mem_ship]*(kac_mem_ship/2.0);
fzy_payda+=(kac_mem_ship/2.0);
fzy_bul=fzy_pay/fzy_payda;

/*fzy_bul=(dog_deg[0]*fzy_uy_deg[0]+dog_deg[1]*fzy_uy_deg[1]+

dog_deg[2]*fzy_uy_deg[2]+dog_deg[3]*fzy_uy_deg[3]+2.0*dog_deg[4])/

(fzy_uy_deg[0]+fzy_uy_deg[1]+fzy_uy_deg[2]+fzy_uy_deg[3]+2.0);
 */
//printf("x0=%lf      x1=%lf      x2=%lf      x3=%lf      X4=%lf      x5=%lf
",x_nokta[0],x_nokta[1],x_nokta[2],x_nokta[3],
//x_nokta[4],x_nokta[5]);
//printf("f0=%lf                f1=%lf                f2=%lf                f3=%lf
",fzy_uy_deg[0],fzy_uy_deg[1],fzy_uy_deg[2],fzy_uy_deg[3]);
fprintf(dosya,"%lf\n",fzy_bul);
if(transfer_mi)
{
fprintf(hata_dos,"%lf\n",fabs(fzy_bul-gec_dizi[(int)bul_x]));
```

```
 esk_hata+= fabs(fzy_bul-gec_dizi[(int)bul_x]);
 esk_h_top+= pow(fabs(fzy_bul-gec_dizi[(int)bul_x]),2.0);
 }
}//döngü.

if(transfer_mi)
 {
        esk_hata/=(son_deg-ilk_deg);
        esk_h_top/=(son_deg-ilk_deg);
        //
        std_sap_oku("c:\\user\\bekir\\st.dat",is_std);

        esk_h_top=sqrt(esk_h_top);

        esk_h_top=20.0*log10(is_std/esk_h_top);

        sprintf(esk_ileti,"%lf",esk_hata);
        sprintf(esk_ile_top,"%lf",esk_h_top);

        fclose(hata_dos);
 }
fclose(dosya);
 }

void new_fzy_sonuc(const char *gd_adi,int ilk_deg,int son_deg,
                        int noktasayi,int kac_mem_ship,int fzy_kural,int transfer_mi)
{
 int i,d_nk;
 double bul_x,basl,x_nokta[5],xg[10],xo[10],xs[10],fzy_uy_deg[10],fzy_bul,
                                yen_h_top,is_std;
 deger dog_deg;
```

```
FILE *dosya;
FILE *hata_dos;
yen_hata=0.0;yen_h_top=0.0;

if(transfer_mi)
 {
       if((hata_dos=fopen("c:\\user\\bekir\\yenih.dat","w"))==NULL) exit(1);
       dos_text_foroku_ac(dos_ism,0,boyut,artim,gec_dizi);
 }

if((dosya=fopen(gd_adi,"w"))==NULL) exit(1);
//gözönünde bulunacak degerler
//i sinirları belirlenecek

switch(fzy_kural){
 case 6:d_nk=1;break;
 case 8:d_nk=2;break;
 case 10:d_nk=3;break;
 case 12:d_nk=4;break;
 case 14:d_nk=5;break;
 }
flushall();
//cizim için
ciz_deg=ilk_deg+d_nk*(noktasayi-1);
for(i=(ilk_deg+d_nk*(noktasayi-1));i<(son_deg+d_nk*(noktasayi-1));i++){
 basl=i;
 bul_x=2.0*(noktasayi-1)+basl;

 new_fzy_dogru_snc(bul_x,noktasayi,basl,dog_deg,x_nokta);

 xg[0]=basl;
```

```
xo[0]=x_nokta[0];
xs[0]=x_nokta[1];

xg[1]=x_nokta[0];
xo[1]=x_nokta[1];
xs[1]=x_nokta[2];

xg[2]=x_nokta[1];
xo[2]=x_nokta[2];
xs[2]=x_nokta[3];

xg[3]=x_nokta[2];
xo[3]=x_nokta[3];
xs[3]=x_nokta[4];

xg[4]=x_nokta[3];
xo[4]=x_nokta[4];
xs[4]=2.0*x_nokta[4]-x_nokta[3];

fzy_mem_ship_snc(kac_mem_ship,xg,xo,xs,bul_x,fzy_uy_deg);

if( (bul_x >= x_nokta[0])&&(bul_x < x_nokta[1]))
      fzy_bul=(dog_deg[3]*fzy_uy_deg[0]+dog_deg[2]*fzy_uy_deg[1]);

else if( (bul_x >= x_nokta[1])&& (bul_x < x_nokta[2]) )
      fzy_bul=(dog_deg[0]*fzy_uy_deg[1]+dog_deg[3]*fzy_uy_deg[2]);

else if( (bul_x >= x_nokta[2])&&(bul_x < x_nokta[3]) )
      fzy_bul=(dog_deg[4]*fzy_uy_deg[2]+dog_deg[1]*fzy_uy_deg[3]);

else if( (bul_x >= x_nokta[3])&&(bul_x <= x_nokta[4]) )
```

```
        fzy_bul=(dog_deg[2]*fzy_uy_deg[3]+dog_deg[4]*fzy_uy_deg[4]);

 fprintf(dosya,"%lf\n",fzy_bul);

 if(transfer_mi)
 {
        fprintf(hata_dos,"%lf\n",fabs(fzy_bul-gec_dizi[(int)bul_x]));
        yen_hata+=fabs(fzy_bul-gec_dizi[(int)bul_x]);
        yen_h_top+=pow(fabs(fzy_bul-gec_dizi[(int)bul_x]),2.0);
 }
 }//döngü
 fclose(dosya);

 if(transfer_mi)
 {
 yen_hata/=((son_deg+d_nk*(noktasayi-1))-(ilk_deg+d_nk*(noktasayi-1)));
 yen_h_top/=((son_deg+d_nk*(noktasayi-1))-(ilk_deg+d_nk*(noktasayi-1)));
 std_sap_oku("c:\\user\\bekir\\st.dat",is_std);
 yen_h_top=sqrt(yen_h_top);
 yen_h_top=20.0*log10(is_std/yen_h_top);
 sprintf(yen_ileti,"%lf",yen_hata);
 sprintf(yen_ile_top,"%lf",yen_h_top);
 fclose(hata_dos);
 }

}
//sin fonk elde ediliyor.frekans ve örnek degerine göre.
void sin_fonk_hesap(const char *d_adi,double f)
{
 FILE *dosya;
```

```
 double sonuc;
 int i;

 if((dosya=fopen(d_adi,"w"))==NULL) exit(1);

        flushall();
        //frekansa bağlı sin fonk. bulunuyor
        for(i=0;i<boyut;i++)
         {
          sonuc=sin(2.0*M_PI*(1.0/11000.0)*i*f);
          fprintf(dosya,"%lf\n",sonuc);
         }

 fclose(dosya);
}

//dosyadan okuma yapılıp, okunanan değerlerin karesi alınıp bütün değerler
//toplandıktan sonra dosyaya yazılıyor.
void fonk_topla(const char *gd_adi,const char *sd_adi,int i_deg,int s_deg)
{
 FILE *dosya,*dosya1;

 double sonuc,ara;

 int i;

if((dosya=fopen(gd_adi,"w"))==NULL) exit(1);
if((dosya1=fopen(sd_adi,"r"))==NULL) exit(1);

 flushall();
 ara=0.0;
```

```
 for(i=0;i<s_deg;i++)
        {
         fscanf(dosya1,"%lf\n",&sonuc);
         if( i >= i_deg)
          ara+= pow(sonuc,2.0);
        }
 fprintf(dosya,"%lf\n",ara);
 fcloseall();
}

//transfer fonksiyonu bulunuyor. i=0 ise dosya yeniden olusturuluyor.
//değilse dosyaya ekleme yapiliyor.
void suz_kar(const char *sd_adi,const char *fzy_d,const char *fnk_d,int i)
{
 FILE *ods,*fds,*sds;

 double deger,cikis,sonuc;

 if((ods=fopen(fnk_d,"r"))==NULL) exit(1);
 if((fds=fopen(fzy_d,"r"))==NULL) exit(1);

 if(i==0)
 {
        if((sds=fopen(sd_adi,"w"))==NULL)
         exit(1);
 }
 else
        if((sds=fopen(sd_adi,"a"))==NULL) exit(1);

flushall();
fscanf(ods,"%lf",&deger);
```

```
fscanf(fds,"%lf",&cikis);
sonuc=10.0*log10(cikis/deger);
fprintf(sds,"%lf\n",sonuc);
fcloseall();
}

//old_fuzzy ve new_fuzzy için transfer fonksiyonlari bulunuyor.
void transfer_fonk(int sinir,int son_deger,double F,double artim,
                                        int  noktasayi,int  kac_mem_ship,int
fzy_kural)
{
 int i,bas_deg,son_deg,d_nokta;
 //old fuzzy ve new fuzzy için sinir belirlenmelidir.
 for(i=0;i<son_deger;i++)
 {
        sin_fonk_hesap("c:\\user\\bekir\\fonk.dat",F);
        F+=artim;
        dos_text_foroku_ac("c:\\user\\bekir\\fonk.dat",0,boyut,1,dizi);

//new fuzzy membership sayisi herzaman 5 ve fuzzy kural değişmez;
//old fuzzy  noktasayi=3;kac_mem_ship=4; fzy_kural=6;
//for new fzy 2.0*(noktasayi-1)+basl;
//for old fuzzy  (3.0+d_nokta)*(noktasayi-1)+basl;

 old_fzy_sonuc("c:\\user\\bekir\\f1fonk.dat",0,600,noktasayi,kac_mem_ship,

                                        fzy_kural,0);
new_fzy_sonuc("c:\\user\\bekir\\f2fonk.dat",0,600,noktasayi,5,fzy_kural,0);

        switch(fzy_kural){
         case 6:d_nokta=0;break;
```

```
        case 8:d_nokta=1;break;
        case 10:d_nokta=2;break;
        case 12:d_nokta=3;break;
        case 14:d_nokta=4;break;
}
      bas_deg=(3.0+d_nokta)*(noktasayi-1);
      //bul_deg=bas_deg;
      son_deg=sinir+(3.0+d_nokta)*(noktasayi-1);

      fonk_topla("c:\\user\\bekir\\sfonk.dat",

      "c:\\user\\bekir\\fonk.dat",bas_deg,son_deg);

      fonk_topla("c:\\user\\bekir\\s1fonk.dat",
                                    "c:\\user\\bekir\\f1fonk.dat",0,sinir);

      fonk_topla("c:\\user\\bekir\\s2fonk.dat",
                                    "c:\\user\\bekir\\f2fonk.dat",0,sinir);

      suz_kar("c:\\user\\bekir\\sefonk.dat",
                    "c:\\user\\bekir\\s1fonk.dat",
                    "c:\\user\\bekir\\sfonk.dat",i);

      suz_kar("c:\\user\\bekir\\syfonk.dat",
                    "c:\\user\\bekir\\s2fonk.dat",
                    "c:\\user\\bekir\\sfonk.dat",i);
 }//for
}
////
//random bir değer üretiliyor.
float  ra(void)
```

```
        {
        float ran, q, w;
        int a;
        q = random(10000);
        w = q / 10000;
        a = random(2);
        if ((a == 1)) ran = 1 * w;
         if ((a == 0)) ran = -1 * w;
        return ran;
}

//gauss üretiliyor.
float gasdev(int  iset)
{
        float  v1, v2, gset, r, fac;
        do{
                v1 = 2.0* ra() - 1.0;
                v2 = 2.0* ra() - 1.0;
                r = v1*v1 + v2*v2;
        }while(r >= 1.0);

        fac = sqrt(-2.0 * log(r) / r);
        gset = v1 * fac;
        if ((iset == 0))
        {
        iset=1;
        return v2 * fac;
        }
        else
        {
                iset=0;
```

```
            return gset;
        }
}

//gürültü üretiliyor.
void gurultu(int son_deg,double gir_std_sap,double &bul_std_sap)
{
        double snr,sigma,ort_deg,kare_ort;
        int sett,k;
        float ss;

        snr=20.0*log10(1.0/(sqrt(2.0)*gir_std_sap));
        sigma=1.0/(sqrt(2.0)*exp(snr*log(10.0)/20.0));
        sett=0;

        randomize();
        for (k = 0; k <son_deg; k++)
                {
                        ss = gasdev(sett);
                        dizi[k]=ss*sigma;
                }

               ort_deg=0.0;
               kare_ort=0.0;

        for (k = 0; k <son_deg; k++)
               {
               ort_deg+=dizi[k];
               kare_ort+=dizi[k]*dizi[k];
               }
       ort_deg=ort_deg/(double)son_deg;
```

```
        kare_ort=kare_ort/(double)son_deg;
        bul_std_sap=sqrt(kare_ort-ort_deg*ort_deg);
        //dosya  yazma modunda açılıp dosyaya yazılacak
        dos_text_foryaz_ac("c:\\user\\bekir\\n.dat",0,son_deg,1);
}

//standart sapma bulunup ilgili dosyaya yaziliyor.
void std_sapma_bul(const char *g_dos_adi,const char *c_dos_adi,
                        int son_deg)
{
 FILE *dosya;
 double s1,s2;
 s1=s2=0.0;

 dos_text_foroku_ac(g_dos_adi,0,son_deg,1,dizi);

 if((dosya=fopen(c_dos_adi,"w"))==NULL) exit(1);

 flushall();
 for(int i=0;i<son_deg;i++)
 {
        s1+=pow(dizi[i],2.0);
        s2+=dizi[i];
 }
 s1/=son_deg;
 s2/=son_deg;
 fprintf(dosya,"%lf\n", sqrt(s1 - s2*s2));
 fclose(dosya);
}
//standart sapma dosyadan okuyor.
void std_sap_oku(const char *g_dos_adi,double &oku_std_sapma)
```

```
{
 FILE *dosya;
 flushall();
 if((dosya=fopen(g_dos_adi,"r"))==NULL) exit(1);
        fscanf(dosya,"%lf",&oku_std_sapma);
 fclose(dosya);
}


void g_ekle(int g_deg,int s_deg,double db,int deger)
 {
 double is_std,bul_std,sonuc,gur_std;

 std_sapma_bul(dos_ism,"c:\\user\\bekir\\st.dat",boyut);
 std_sap_oku("c:\\user\\bekir\\st.dat",is_std);

 sonuc=is_std/pow(10.0,(db/20.0));

 gurultu(boyut,sonuc,bul_std);

// suzgecleme();

 std_sapma_bul("c:\\user\\bekir\\n.dat","c:\\user\\bekir\\st_gur.dat",boyut);
 std_sap_oku("c:\\user\\bekir\\st_gur.dat",gur_std);

// sprintf(top_ileti,"%lf",is_std);
// strcat(top_ileti," ");
// sprintf(esk_ileti,"%lf",gur_std);

// strcat(top_ileti," ");
// strcat(top_ileti,esk_ileti);
```

```
// strcat(top_ileti," ");

 dos_text_forekle(dos_ism,"c:\\user\\bekir\\n.dat",

"c:\\user\\bekir\\deger1g.dat",g_deg,s_deg,deger);


 }

 //noktasayi=3,5,7,9,11,...
 //kac_mem_ship=4,6,8 fuzzy_kural=kac_mem_ship+2
 //dizide kaç atlama yapılacak artim=1(atlama yok),2(bir atlamalı),3,4,5,6....
 //yeni yöntem icin üyelik fonk.(mem_ship) adedi 5 ve sabititr.
 //eski yöntemde degisebilir.(4+1=5,6+1=7,8+1=9)Bir üyelik fonk. hep
 //1(bir) degerini  aldigi için üyelik fonksiyonu degeri hesaplanmiyor
 //dizinin 0 ile 199.cu elemanina 10 db(gur_db[0]) ekleniyor
 //dizinin 200 ile 399.cu elemanina 15 db(gur_db[1]) ekleniyor
 //gürültü ekleniyor.bas_deg,bitis_deg,db deg,dosya açılışı durumu
 //new fuzzy membership sayisi herzaman 5 ve fuzzy kural değişmez;
 //transfer fonksiyonu bulunuyor
 //sinir=100;son_deger=70;Fre=10.0;artim=50
 //transfer_fonk(100,35,10.0,100.0,noktasayi,kac_mem_ship,fzy_kural);
//*********
void kalman(int noktasayi,int fzy_kural,int ilk_deg,int sinir)
{
 FILE *kal_dos,*hata_dos;
 int bas_deg,son_deg;
 double dizi_top,dizi_top_kare,kalm,x_once,gur_std,kal_hata,d_nokta;
 int i;
 dizi_top=0;dizi_top_kare=0;kal_hata=0.0;gur_std=0.0;
```

```
        switch(fzy_kural){
         case 6:d_nokta=0;break;
         case 8:d_nokta=1;break;
         case 10:d_nokta=2;break;
         case 12:d_nokta=3;break;
         case 14:d_nokta=4;break;
}

        bas_deg=(3.0+d_nokta)*(noktasayi-1);
        son_deg=sinir+(3.0+d_nokta)*(noktasayi-1);

dos_text_foroku_ac(dos_ism,0,boyut,artim,dizi);
dos_text_foroku_ac("c:\\user\\bekir\\deger1g.dat",0,boyut,artim,gec_dizi);

hata_dos=fopen("c:\\user\\bekir\\kalmh.dat","w");
kal_dos=fopen("c:\\user\\bekir\\kalman.dat","w");

for(i=0;i<5;i++)
        gur_std+=gur_db[i];
gur_std/=5.0;
gur_std=pow(gur_std,2.0);

for(i=ilk_deg + bas_deg;i<son_deg;i++)
 dizi_top+=dizi[i];
dizi_top/=(son_deg-bas_deg);

for(i=ilk_deg + bas_deg;i<son_deg;i++)
 dizi_top_kare+= pow( (dizi[i]-dizi_top),2.0);
dizi_top_kare/=(son_deg-bas_deg);
```

```
x_once=dizi_top;

for(i=ilk_deg + bas_deg;i<son_deg;i++)
{
        kalm=x_once + ( dizi_top_kare / (gur_std+2.0*i) ) * (gec_dizi[i]-x_once);
        x_once=kalm;
        fprintf(kal_dos,"%lf\n",x_once);
 //kontrol
        fprintf(hata_dos,"%lf\n", fabs(kalm-dizi[i-2]));
        kal_hata+=pow((kalm-dizi[i-2]),2.0);
}

kal_hata/=(son_deg-bas_deg);
sprintf(kal_ileti,"%lf",kal_hata);
fcloseall();
}
//********Alçak Geçiren Süzgeç*********
 void al_gec_fil()
{
 FILE *al_dos;
 double deger,ags_hata,is_std;
 int i;
 ags_hata=0.0;

 dos_text_foroku_ac("c:\\user\\bekir\\deger1g.dat",0,boyut,artim,dizi);
 al_dos=fopen("c:\\user\\bekir\\ags.dat","w");

 for(i=0;i<(boyut-2);i++)
 {
        deger=(dizi[i]+dizi[i+1]+dizi[i+2])/3.0;
        fprintf(al_dos,"%lf\n",deger);
```

```
        ags_hata+=pow((deger-dizi[i+1]),2.0);
 }

 ags_hata/=(boyut-2);
 ags_hata=sqrt(ags_hata);
 std_sap_oku("c:\\user\\bekir\\st.dat",is_std);
 ags_hata=20.0*log10(is_std/ags_hata);
 sprintf(yen_ileti,"%lf",ags_hata);
 fclose(al_dos);
}
//********* Alçak Geçiren Süzgeç******
//*********Gürültü Süzgeçleme islemi******
void suzgecleme()
 {
  float fs,fc,ts,c,c1,c2;
  int oran,n,i;
  FILE *dosya;

  fc=2500.0;
  fs=11000.0;
  oran=(int)fs/fc;
  ts=1.0/fs;
  c=2.0*M_PI*fc*ts;
  c2=c/M_PI;

  for(n=(-5*oran);n<0;n++)
        {
        c1=n*c;
        h[n+5*oran]=h[-n+5*oran]=c2*sin(c1)/c1*(0.54+0.46*cos(n*M_PI/(5*oran)));
        }
 h[5*oran]=c2;
```

```
dos_text_foroku_ac("c:\\user\\bekir\\n.dat",0,boyut,artim,dizi);

cik_deg[0]=dizi[0]*h[0];

for(n=1;n<boyut;n++)
 {
        cik_deg[n]=0.0;
        if(n<(10*oran))
         {
          for(i=0;i<n;i++)
                cik_deg[n]+=h[i]*dizi[n-i];
         }
        if((n>=(10*oran)) && ((n-boyut)<=0))
         {
          for(i=0;i<=(10*oran);i++)
                cik_deg[n]+=h[i]*dizi[n-i];
         }
        if((n>=(10*oran)) && ((n-boyut)>0))
         {
          for(i=(n-boyut);i<=(10*oran);i++)
                cik_deg[n]+=h[i]*dizi[n-i];
         }
        }

dosya=fopen("c:\\user\\bekir\\n.dat","w");
for(i=0;i<boyut;i++)
 fprintf(dosya,"%lf\n",cik_deg[i]);
fclose(dosya);
}
```

```
void cizim(const char *dosya_adi,HPEN kalem,int artim)
{
 int i,ek=0;

 dos_text_foroku_ac(dosya_adi,0,500,artim,dizi);

 if(stricmp(dosya_adi,"c:\\user\\bekir\\deg.dat")==0)ek=6;
 if(stricmp(dosya_adi,"c:\\user\\bekir\\deger1g.dat")==0)ek=6;
 if(stricmp(dosya_adi,"c:\\user\\bekir\\ags.dat")==0)ek=5;
 SelectObject(memdc,kalem);

 MoveToEx(memdc,2*(20+artim-ek),100+2.0*dizi[0],NULL);

 for(i=0;i<500;i++)
 LineTo(memdc,2*(i+20+artim-ek),100+2.0*dizi[i]);
}

void cizimt(const char *dosya_adi,HPEN kalem)
{int i;

 dos_text_foroku_ac(dosya_adi,0,500,1,dizi);
 SelectObject(memdc,kalem);

 MoveToEx(memdc,200*log10(10),40-8*dizi[0],NULL);
 for(i=0;i<35;i++)
        LineTo(memdc,200*log10(4*i+10),40-8*dizi[i]);
 }


int PASCAL WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
        LPSTR lpszArgs, int nWinMode)
```

```
{
    HWND      hwnd;
    MSG       msg;
    WNDCLASS  wcl;

        wcl.style       = 0;
        wcl.lpfnWndProc  = WindowFunc;
        wcl.cbClsExtra   = 0;
        wcl.cbWndExtra   = 0;
        wcl.hInstance    = hThisInst;
        wcl.hIcon        = LoadIcon(hThisInst, "MYICON");
        wcl.hCursor      = LoadCursor(NULL, IDC_ARROW);
        wcl.hbrBackground = GetStockObject(WHITE_BRUSH);
        wcl.lpszMenuName  = "MYMENU";
        wcl.lpszClassName = szWinName;

                if(!RegisterClass(&wcl)) return 0;

hwnd = CreateWindow(szWinName,  // window class name
        "Bulanık Mantık Uygulaması",      // window caption
        WS_OVERLAPPEDWINDOW,      // window style
        CW_USEDEFAULT,          // initial x position
        CW_USEDEFAULT,          // initial y position
        CW_USEDEFAULT,          // initial x size
        CW_USEDEFAULT,          // initial y size
        HWND_DESKTOP,           // parent window handle
        NULL,                   // window menu handle
        hThisInst,              // program instance handle
        NULL);                  // creation parameters

hInst=hThisInst;
```

```
        ShowWindow(hwnd, nWinMode);
        UpdateWindow(hwnd);

        while (GetMessage(&msg, NULL, 0, 0)) {
                TranslateMessage(&msg);
                DispatchMessage(&msg);

        }
        return msg.wParam;
}

LRESULT CALLBACK WindowFunc(HWND hwnd,UINT message,WPARAM
wParam,
                                                        LPARAM lParam)
{
        HDC hdc;
        PAINTSTRUCT paintstruct;

         switch (message)
         {
                case WM_CREATE:
                                maxX=GetSystemMetrics(SM_CXSCREEN);
                                maxY=GetSystemMetrics(SM_CYSCREEN);
                                hdc=GetDC(hwnd);
                                memdc=CreateCompatibleDC(hdc);
                                hbit=CreateCompatibleBitmap(hdc,maxX,maxY);
                                SelectObject(memdc,hbit);
                                hbrush=GetStockObject(WHITE_BRUSH);
                                SelectObject(memdc,hbrush);
                                PatBlt(memdc,0,0,maxX,maxY,PATCOPY);
```

```
                hRedpen=CreatePen(PS_SOLID,1,RGB(255,0,0));
                hGreenpen=CreatePen(PS_SOLID,1,RGB(0,255,0));
                hBluepen=CreatePen(PS_SOLID,1,RGB(0,0,255));
                hGrypen=CreatePen(PS_SOLID,1,RGB(50,0,0));
                hYellowpen=CreatePen(PS_SOLID,1,RGB(255,255,0));
                hWhitepen=CreatePen(PS_SOLID,1,RGB(55,55,55));
                hOrangepen=CreatePen(PS_SOLID,1,RGB(222,155,33));
                hBrownpen=CreatePen(PS_SOLID,1,RGB(113,49,33));

                hOldpen=SelectObject(memdc,hRedpen);
                SelectObject(memdc,hOldpen);
                ReleaseDC(hwnd,hdc);
                break;

        case WM_COMMAND:

                switch(LOWORD(wParam))
                {
                 case CM_DEGERGIRDI:

DialogBox(hInst,"MYDB1",hwnd,(DLGPROC)DialogG);
                        InvalidateRect(hwnd,NULL,1);

                        if(secimOK==1)
                        {
                        //suzgecleme();
                        g_ekle(0,200,gur_db[0],0);
                        g_ekle(200,400,gur_db[1],1);
                        g_ekle(400,600,gur_db[2],1);
                        g_ekle(600,800,gur_db[3],1);
```

```
                    g_ekle(800,1000,gur_db[4],1);

    dos_text_foroku_ac("c:\\user\\bekir\\deger1g.dat",0,1000,artim,dizi);

    new_fzy_sonuc("c:\\user\\bekir\\yenif.dat",0,900,noktasayi,5,fzy_kural,1);

    old_fzy_sonuc("c:\\user\\bekir\\eskif.dat",0,900,noktasayi,kac_mem_ship,

                                                    fzy_kural,1);
                    al_gec_fil();
                    *top_ileti='\0';
                    //strcat(top_ileti," ");
                    strcat(top_ileti,yen_ile_top);
                    strcat(top_ileti," ");
                    //strcat(top_ileti,yen_ileti);
                    //strcat(top_ileti," ");
                    strcat(top_ileti,esk_ile_top);
                    strcat(top_ileti," ");
                    strcat(top_ileti,yen_ileti);

    sprintf(gec_dur,"%lf",(gur_db[0]+gur_db[1]+gur_db[2]+gur_db[3]+gur_db[4])/
5.0);
                    strcat(top_ileti," ");
                    strcat(top_ileti,gec_dur);

MessageBox(hwnd,top_ileti,"Y_Gr_Db  E_Gr_Db  Ags_Gr_Db Or_Gr_Db",
            MB_ICONINFORMATION|MB_ICONHAND);

//transfer_fonk(100,35,10.0,100.0,noktasayi,kac_mem_ship,fzy_kural);

                    //kalman(noktasayi,fzy_kural,0,900);
```

```
                    InvalidateRect(hwnd,NULL,1);

cizim("c:\\user\\bekir\\deger1g.dat",hOrangepen,artim);
//cizim("c:\\user\\bekir\\deg.dat",hYellowpen,artim);
                    cizim("c:\\user\\bekir\\ags.dat",hBrownpen,artim);

                    SelectObject(memdc,hOldpen);
                    SelectObject(memdc,hOldbrush);
                    DeleteObject(hbrush);
                    secimOK=0;
                    }
                break;
            case CM_ISARETY:
                InvalidateRect(hwnd,NULL,1);
                cizim("c:\\user\\bekir\\yenif.dat",hRedpen,1);
                SelectObject(memdc,hOldpen);
                SelectObject(memdc,hOldbrush);
                DeleteObject(hbrush);

                break;

                case CM_ISARETE:
                InvalidateRect(hwnd,NULL,1);
                cizim("c:\\user\\bekir\\eskif.dat",hBluepen,1);
                SelectObject(memdc,hOldpen);
                SelectObject(memdc,hOldbrush);
                DeleteObject(hbrush);

                break;
                case CM_HATAY:
```

```
                    InvalidateRect(hwnd,NULL,1);
                    cizim("c:\\user\\bekir\\yenih.dat",hGrypen,1);
                    MessageBox(hwnd,yen_ileti,"Toplamsal Hata",

MB_ICONINFORMATION|MB_ICONHAND);
                    SelectObject(memdc,hOldpen);
                    SelectObject(memdc,hOldbrush);
                    DeleteObject(hbrush);

                   break;

                   case CM_HATAE:
                    InvalidateRect(hwnd,NULL,1);
                    cizim("c:\\user\\bekir\\eskih.dat",hYellowpen,1);
                    MessageBox(hwnd,esk_ileti,"Toplamsal Hata",
                          MB_ICONINFORMATION|MB_ICONHAND);
                    SelectObject(memdc,hOldpen);
                    SelectObject(memdc,hOldbrush);
                    DeleteObject(hbrush);
                   break;


                   case CM_SUZGECY:
                    InvalidateRect(hwnd,NULL,1);
                    cizimt("c:\\user\\bekir\\syfonk.dat",hRedpen);
                    SelectObject(memdc,hOldpen);
                    SelectObject(memdc,hOldbrush);
                    DeleteObject(hbrush);

                   break;
```

```
case CM_SUZGECE:
        InvalidateRect(hwnd,NULL,1);
        cizimt("c:\\user\\bekir\\sefonk.dat",hBluepen);
        SelectObject(memdc,hOldpen);
        SelectObject(memdc,hOldbrush);
        DeleteObject(hbrush);

        break;
case CM_KALI:
        InvalidateRect(hwnd,NULL,1);
        cizim("c:\\user\\bekir\\kalman.dat",hBluepen,1);
        SelectObject(memdc,hOldpen);
        SelectObject(memdc,hOldbrush);
        DeleteObject(hbrush);

        break;
case CM_KALH:
        InvalidateRect(hwnd,NULL,1);
        cizim("c:\\user\\bekir\\kalmh.dat",hRedpen,1);
        MessageBox(hwnd,kal_ileti,"Toplamsal Hata",
        MB_ICONINFORMATION|MB_ICONHAND);
        SelectObject(memdc,hOldpen);
        SelectObject(memdc,hOldbrush);
        DeleteObject(hbrush);

        break;

case CM_SILME:MessageBox(hwnd,"Ekrani
tazeleme","Silme",MB_ICONINFORMATION|MB_OK);
        MoveToEx(memdc,0,0,NULL);
        PatBlt(memdc,0,0,maxX,maxY,PATCOPY);
```

```
                    InvalidateRect(hwnd,NULL,1);
                    break;

                case CM_YAR:

    DialogBox(hInst,"MYDB2",hwnd,(DLGPROC)DialogG);
                    break;
                }
                break;
            case WM_PAINT:
            hdc=BeginPaint(hwnd,&paintstruct);
            BitBlt(hdc,0,0,maxX,maxY,memdc,0,0,SRCCOPY);
            EndPaint(hwnd,&paintstruct);
            break;

            case WM_DESTROY:
                DeleteObject(hRedpen);
                DeleteObject(hGreenpen);
                DeleteObject(hBluepen);
                DeleteObject(hGrypen);
                DeleteObject(hYellowpen);
                DeleteObject(hWhitepen);
                DeleteDC(memdc);
                PostQuitMessage(0);
                break;

        default:
                return DefWindowProc(hwnd, message, wParam, lParam);
 }
 return 0;
}
```

```
BOOL CALLBACK DialogG(HWND hdwnd,UINT message,WPARAM
wParam,LPARAM lParam)
 {
        long icerik;

        switch(message){
              case WM_COMMAND:
                    switch(LOWORD(wParam)){
                     case ID_PB2:EndDialog(hdwnd,0);
                     secimOK=0;
                     return 1;

        case ID_PB3:MessageBox(hdwnd,"Değer Girilmesi","",MB_ICONQUESTION |
MB_OK);
                     secimOK=0;
                     return 1;

                     case ID_PB1:
                     secimOK=1;

status1=SendDlgItemMessage(hdwnd,ID_RB1,BM_GETCHECK,0,0);

status2=SendDlgItemMessage(hdwnd,ID_RB2,BM_GETCHECK,0,0);

status3=SendDlgItemMessage(hdwnd,ID_RB3,BM_GETCHECK,0,0);

status4=SendDlgItemMessage(hdwnd,ID_RB4,BM_GETCHECK,0,0);

status5=SendDlgItemMessage(hdwnd,ID_RB5,BM_GETCHECK,0,0);
```

```
status6=SendDlgItemMessage(hdwnd,ID_RB6,BM_GETCHECK,0,0);

status7=SendDlgItemMessage(hdwnd,ID_RB7,BM_GETCHECK,0,0);

status8=SendDlgItemMessage(hdwnd,ID_RB8,BM_GETCHECK,0,0);

                    if(status1) noktasayi=3;
                    else if(status2) noktasayi=5;
                    else if(status3) noktasayi=7;
                    else if(status4) noktasayi=9;
                    else if(status8) noktasayi=2;

                    if(status5){ kac_mem_ship=4;fzy_kural=6;}
                    else if(status6){ kac_mem_ship=6;fzy_kural=8;}
                    else if(status7) {kac_mem_ship=8;fzy_kural=10;}
                    gur_db[0]=(double)GetDlgItemInt(hdwnd,ID_E1,NULL,0);
                    gur_db[1]=(double)GetDlgItemInt(hdwnd,ID_E2,NULL,0);
                    gur_db[2]=(double)GetDlgItemInt(hdwnd,ID_E3,NULL,0);
                    gur_db[3]=(double)GetDlgItemInt(hdwnd,ID_E4,NULL,0);
                    gur_db[4]=(double)GetDlgItemInt(hdwnd,ID_E5,NULL,0);
                    icerik=GetDlgItemInt(hdwnd,ID_E6,NULL,0);
                    //artim=(int)icerik;
                    artim=1;
                    switch((int)icerik)
                          {
                           case 1:strcpy(dos_ism,"c:\\user\\bekir\\deg.dat");break;
                           case 2:strcpy(dos_ism,"c:\\user\\bekir\\deg1.dat");break;
                           case 3:strcpy(dos_ism,"c:\\user\\bekir\\deg2.dat");break;
                           case 4:strcpy(dos_ism,"c:\\user\\bekir\\deg3.dat");break;
                           default:strcpy(dos_ism,"c:\\user\\bekir\\deg.dat");
```

```
                    }
                EndDialog(hdwnd,0);

                return 1;
            }
        }
        return 0;
}
BOOL CALLBACK DialogY(HWND hdwnd,UINT message,WPARAM
wParam,LPARAM lParam)
 {
        switch(message){
            case WM_COMMAND:
                switch(LOWORD(wParam)){
                case ID_PB4:
                EndDialog(hdwnd,0);
                secimOK=0;
                return 1;
            }
        }
return 0;
}
```

8. ÖZGEÇMİŞ

Bekir DİZDAROĞLU; 1971 yılında Giresun'da doğdu. İlkokulu Giresun Samanlık Kıranı İlkokulu'nda, orta ve liseyi ise Giresun İmam-Hatip Lisesi'nde tamamladı. 1990 yılında Karadeniz Teknik Üniversitesi Elektrik-Elektronik Mühendisliği Bölümüne başladı. 1994 yılında bu bölümden Elektronik Mühendisi olarak mezun oldu ve o yıl ayni üniversitenin Elektronik dalında Yüksek Lisans öğrenimine başladı. 1995 yılında KTÜ Beşikdüzü Meslek Yüksekokulu'nda öğretim elemanı olarak görev yapmaya başladı. Yabancı dil olarak İngilizce bilmektedir.