# Adaptif Süzgeçlerde Farksal Gelişim Algoritması Kullanılarak Gürültü Giderme

# Noise Cancellation Using Differential Evolution Algorithm For Adaptive Filters

*Nalân YİĞİT[1] Nurhan KARABOĞA[2] Burak GÜRER[3]*

Elektrik Elektronik Mühendisliği Bölümü
Mühendislik Fakültesi
Erciyes Üniversitesi, 38090, Talas, Kayseri
e-posta[1]: nalanygt@yahoo.com e-posta[2]: nurhan_k@erciyes.edu.tr
e-posta[3]: burakgurer@gmail.com

## Özetçe

Bu çalışmada, gürültü giderme için adaptif sonlu darbe yanıtlı(FIR) süzgeçlerin tasarımında Farksal Gelişim algoritmasının kullanılması anlatılmıştır. Geliştirilen gürültü gidericinin farklı SNR değerlerindeki performansı incelenmiştir.

## Abstract

In this paper, Differential Evolution Algorithm applied to noise cancellation using adaptive finite impulse response(FIR) filters. The performance of the improved noise canceller for different SNR values is analised.

## 1. Giriş

Gürültüyle bozulmuş bir işareti tahmin etmenin en genel yolu, nispeten değişmemiş işarete izin verirken gürültüyü bastırmaya yönelik bir süzgeç vasıtasıyla birleşik işareti geçirmektir. Buradaki belirtilen amaçlar için kullanılan süzgeçler sabit yada adaptif olabilirler. Sabit süzgeçlerin tasarımı, hem işaretin hem de gürültünün önceki bilgilerine dayalı olmalıdır fakat adaptif süzgeçler, kendi parametrelerini otomatik olarak ayarlama kabiliyetine sahiptir ve tasarımları işaretin ya da gürültünün özelliklerinin önceki bilgilerine çok az ihtiyaç duyularak veya hiç gerek duyulmaksızın yapılmaktadır. Gürültü giderme birçok uygulamada çok fazla avantaja sahip olan optimal süzmenin bir çeşididir. İşaretin algılanamadığı ya da zayıf olduğu gürültü alanındaki noktalara yerleştirilen bir veya daha fazla algılayıcıdan elde edilmiş yardımcı ya da referans giriş kullanılır. Giriş süzülür ve işaret ve gürültünün her ikisini de içeren ana(primary) girişten çıkartılır. Sonuç olarak, gürültü zayıflatılır ve giderici tarafından elimine edilir. Bununla birlikte, gürültü indirgeme uygulaması eğer süzmede ve çıkarma uygun bir adaptif süzgeç tarafından kontrol edilirse, birçok durumda işaretin bozulması veya çıkış gürültü seviyesinin artması gibi küçük risklerle başarılabilir. Adaptif gürültü gidermenin uygulanabildiği durumlarda, direkt süzmeyle başarılması zor veya imkansız olan gürültü giderme belli derecede başarılabilir[1]. Bununla birlikte olumsuzlukların üstesinden gelebilmek için evrimsel algoritmalar da gürültü gidermede kullanılmaktadırlar[2-4]. Bu çalışmada, basit ve güçlü popülasyon tabanlı bir algoritma olduğu için evrimsel algoritmalar içerisinde yer alan Farksal Gelişim (FG- Differential Evolution) algoritması kullanılmıştır. Farksal Gelişim algoritmasının basit yapısı, kullanım kolaylığı, hızı ve dinçliği en önemli avantajları arasında yer almaktadır.

Bilgisayarlar, performanslarının gelişimi ile evrimsel algoritmalara dayanan optimizasyon metotlarının yeni bir sınıfını kullanarak bazı teknik problemlerin çözümüne izin vermektedirler. Farksal Gelişim algoritması ilk olarak 1995 yılında K. Price tarafından ortaya konmuştur[4]. FG algoritması basit ama güçlü popülasyon tabanlı bir algoritmadır. Özellikle bütünüyle düzenlenmiş uzayda tanımlı ve gerçek değerli tasarım parametrelerini içeren fonksiyonları, küresel olarak optimize etmek amacıyla kullanılan bir direkt araştırma algoritmasıdır[5]. Gürültü olduğunda, Adaptif süzgeçlerin ağırlıkları adaptif olarak ayarlanmaktadır.

İkinci ve Üçüncü bölümlerde sırasıyla Adaptif Gürültü Gidermenin ve FG Algoritmasının kısa bir özeti verilmektedir. Dördüncü bölümde Adaptif süzgeçlerde gürültü gidermede farklı SNR(Signal to Noise Ratio- İşaret Gürültü Oranı) değerleri için algoritmanın nasıl uygulandığı konusu anlatılarak elde edilen simülasyon sonuçları sunulmaktadır.

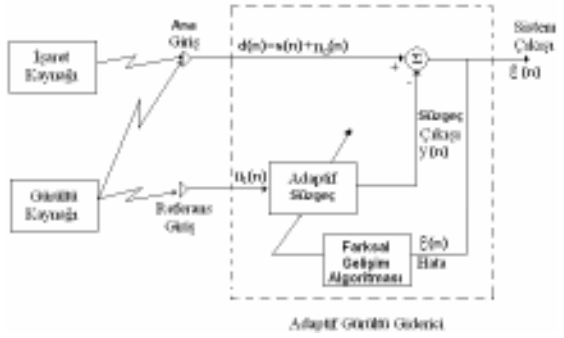
## 2. Adaptif Gürültü Giderme

Adaptif Süzgeç, otomatik ayar yapabilen sayısal süzgeçlerdir. Bu süzgeçler giriş işaretlerine göre değişebilirler. Adaptif Süzgeç, değişken işaret durumlarına karşılık süzgeç karakteristiklerinin değişimini gerektiren uygulamalarda kullanılırlar[1].

Bilgi işareti s(n), ilintisiz bir gürültü olan $n_0$ ile toplanan işareti alan algılayıcıya bir kanal üzerinden iletilir. Toplam bilgi işareti ve gürültü, $(s+n_0)$,

gidericinin öncelikli girişini oluşturur. İkinci bir algılayıcı, bilgi işaretiyle ilintili olmayan ama $n_0$ gürültüsüyle ilintili olan $n_1$ gürültüsünü alır. Bu algılayıcı, gürültü gidericiye referans girişini sağlar. $n_1$ gürültüsü, $n_0$'ın yakın bir kopyası olan bir y çıkışını üretmek için süzülür. Bu çıkış, $s+n_0$ girişinden $s+n_0$-y sistem çıkışını üretmek için çıkartılır. Adaptif süreçte kullanılan hata işareti uygulamanın doğasına bağlıdır. Gürültü giderme sistemlerinde amaç sistem çıkışı olan s+ $n_0$-y' yi üretmektir ki bu s işaretine, en küçük kareler hassaslığında en uygun olandır. Bu amaç, sistem çıkışını adaptif süzgece geri besleyerek ve tüm sistem çıkış gücünü küçültmek için adaptif bir algoritma üzerinden süzgeci ayarlayarak başarılmaktadır. $n_0$ gürültüsünün veya s işaretinin daha önceki bazı bilgileri düşünülebilir ve $n_1$'e süzgeç tasarlanmadan önce veya gürültü giderme işareti y'yi üretmeye adapte olmadan önce gerek duyulabilir. İster istatistiksel ister deterministik olsun s, $n_0$ veya $n_1$ veya bağıntıları daha önce bilinen bir bilgiye ihtiyaç duymazlar. Şekil 1'de farksal gelişim algoritması kullanılarak tasarlanan adaptif gürültü gidericinin yapısı verilmiştir.

s(n) = Bilgi işareti
$n_o$(n) = Gürültü işareti
$n_1$(n) = Referans işaret
ε(n) = Hata işareti




*Şekil 1*: Farksal gelişim algoritmasının adaptif gürültü gidericide kullanılması

Şekil 1'den sistem çıkışı için

$$\varepsilon = s + n_0 - y \qquad (1)$$

yazılabilir. Buradan görüleceği gibi, $y=n_0$ olduğunda ε=s 'ye eşit olacaktır.

## 3. Farksal Gelişim Algoritması

Farksal gelişim algoritması küresel optimizasyon için basit ama güçlü popülasyon tabanlı bir algoritmadır[6]. Algoritmanın önemli parametreleri: NP (Number of Population-popülasyon büyüklüğü), CR (Crossover Rate-çaprazlama sabiti), F (Scaling Factor-ölçekleme faktörü) olarak sayılabilir. FG'da, NP adet çözüm vektörünün bir popülasyonu başlangıçta rasgele meydana getirilir. Bu popülasyon mutasyon, çaprazlama ve seçme (selection) operatörleri uygulanarak başarılı bir şekilde geliştirilir[7-8]. Farksal Gelişim algoritmasının temel adımları aşağıdaki gibi verilebilir:

*Başlangıç popülasyonunun oluşturulması*
*Değerlendirme*
***Tekrarla***
*Mutasyon*
*Yeniden birleştirme*
*Değerlendirme*
*Seçme*
***D*urdurma kriteri sağlanıncaya kadar**

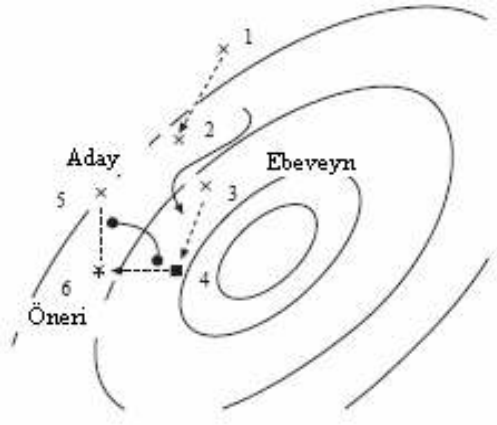
Farksal gelişim algoritması, diğer Evrimsel Algoritmalardan mutasyon ve yeniden birleştirme(rekombinasyon) aşamalarında farklılık göstermektedir. Farksal gelişim algoritması popülasyonu karıştırmak için, çözüm vektörleri arasındaki ağırlıklandırılmış farkları kullanmaktadır[9].

$$\mathbf{u}_{i;G+1} = x_{i,G} + K\cdot (x_{r3};G\_ x_{i,G}) + F \cdot (x_{r1,G} - x_{r2,G})$$

Rasgele seçilen $r_1; r_2; r_3 \in \{1, 2, ..., n\}$;

$$r_1 \neq r_2 \neq r_3 \neq i \qquad (2)$$

Bu çalışmada kullanılan Farksal gelişim algoritması, DG/current-to-rand/1 (Eşitlik 2) olarak bilinir ve rotasyonel olarak değişimsizdir (invariant)[9]. Bir Farksal Gelişim Algoritması popülasyonu başlangıç parametre sınırları içerisinde rasgele başlatılır. Her bir G iterasyonunda, popülasyon pertürbasyona uğrar. Birbirinden farklı üç birey ya da x ile gösterilen çözüm vektörleri popülasyondan rasgele seçilir. K katsayısı $x_{r3,G}$ ve mevcut birey $x_{i,G}$ arasında meydana gelen birleşmenin seviyesini temsil eder. F katsayısı ise $x_{r1,G}$ - $x_{r2,G}$ vektör farkından ileri gelen adım büyüklüğünün ölçeklenmesini gösterir.




*Şekil 2.* FG algoritmasında yeni bir çözümün elde edilmesi.

Yeni bir çözümün üretilmesi Şekil 2'de gösterilmiştir. Şekil 2'ye göre; iki popülasyon üyesi (1,2) arasındaki

fark üçüncü popülasyon üyesine (3) ilave edilmekte ve sonuç (4), aday üyeyle (5) çaprazlama işlemine tutulmaktadır. Böylece yeni çözüm (6) elde edilmektedir. Şayet yeni çözüm, adaydan daha iyiyse adayla yer değiştirmektedir.

### *Kodlama*

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

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

### *Mutasyon*

FG algoritması, bir amaç vektörüne mutasyon işlemini, rastgele seçilmiş amaç vektörler çiftinin ağırlıklaştırılmış farkının bu amaç vektörüne ilave edilmesiyle gerçekleştirir. FG algoritması, gerçek parametre optimizasyonuna rastgele amaç vektörlerinin doğrusal kombinasyonlarını örnekleyen, mutasyon ve yeniden birleştirme gibi iki basit operasyon kullanan normal gelişime dayalı bir yaklaşımdır.

### *Seçme*

Seçme işlemi, yeni üretilen vektörlerin hangi şartlar altında popülasyona girebileceğini tanımlayan bir kriterdir. FG algoritmasının seçme işleminde, yeni üretilen vektör ebeveynine göre daha gelişmiş veya en azından aynı gelişme seviyesinde değilse ebeveyn vektör en az bir jenerasyon daha popülasyonda kalmaya devam etmekte ve başka bir vektörle yer değiştirmemektedir.

### *Yeniden Birleştirme*

Yeniden birleştirme veya çaprazlama işleminin amacı, var olan amaç vektör parametrelerinden faydalanarak yeni vektörleri oluşturmak suretiyle araştırmanın başarılı olması için yardımcı olmaktır. Yeniden birleştirme etkisi üzerine oldukça ciddi çalışmalar ve tartışmalar mevcuttur. Ama kesin olan bir sonuç, uniform yeniden birleştirme işleminin araştırmanın hesaplama maliyetini önemli miktarda artırdığıdır. Ancak araştırmaya önemli ölçüde hız kazandırmaktadır.

## 4. Simülasyon Sonuçları

Simülasyonlar ikinci dereceden adaptif FIR süzgeçler için gerçekleştirilmiştir. Bu çalışma için Eşitlik 3 ile verilen işaretler kullanılmıştır. N, örnekleme periyodu'dur ve 500 olarak alınmıştır.

$$s(n) = \sin(50\pi n/N) \qquad n=1,2,\ldots,1000$$
$$d(n) = s(n) + n_0(n) \tag{3}$$

Eşitlik 3'de verilen s(n) bilgi işaretine, gürültü $n_0(n)$ işareti eklenmiştir. $n_0(n)$ normal dağılımlı rasgele gürültü işaretidir. d(n), SNR değerleri 8,10 ve 13dB olarak, algoritmanın performansını kıyaslamak amacıyla ayrı ayrı ele alınmış işarettir. Referans işaret $n_1(n)$, $n_0(n)$ ile ilintili olan bir gürültü işaretidir.

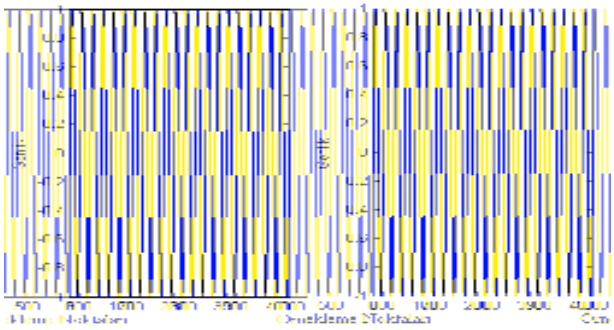
Referans işareti $n_1(n)$ süzgeç girişinde verilmektedir, d(n) olarak gösterilen giriş işareti de süzgeç çıkışıyla karşılaştırılarak hatanın elde edilmesini sağlayan işaret olarak kullanılmaktadır. Bu çalışmada, hata analizi için En Küçük Karesel Hata (LMS-Least Mean Square) kullanılmıştır. Burada elde edilecek sonuç, süzgeç çıkışı y(n) ile ana(primary) giriş işareti d(n) arasındaki fark olarak elde edilmektedir. Bu çalışmada hatayı minimize etmek için LMS (Least Mean Square-En Küçük Karesel Hata) hata fonksiyonu kullanılmıştır. En Küçük Karesel Hata fonksiyonu, arzu edilen işaretten tasarlanan süzgecin çıkışının çıkarılarak elde edilmiş olan hataların kareleri alınarak toplanması ve toplamın karekökünün minimize edilmesi amacını taşımaktadır.

Süzgeç tasarımı için, Farksal gelişim algoritması kullanılarak çıkışla arzu edilen işaret arasındaki hata belirli bir değere minimize edilene kadar süzgeç parametreleri ayarlanmıştır. Tablo-1'de, bu çalışmada kullanılan algoritmanın kontrol parametreleri verilmiştir.

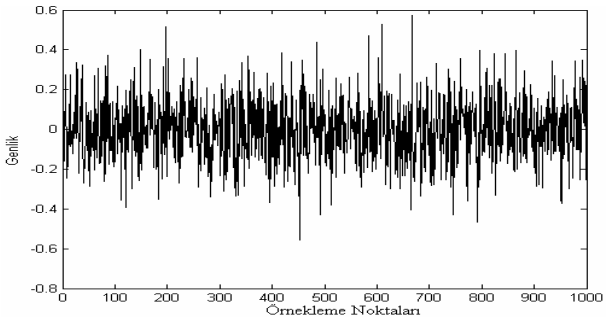
*Tablo-1:* DG'nin kontrol parametreleri

| | |
|---|---|
| Popülasyon büyüklüğü | 100 |
| Çaprazlama oranı | 0.8 |
| Birleşme faktörü | 0.4 |
| Ölçekleme faktörü | 0.5 |
| Jenerasyon sayısı | 1000 |

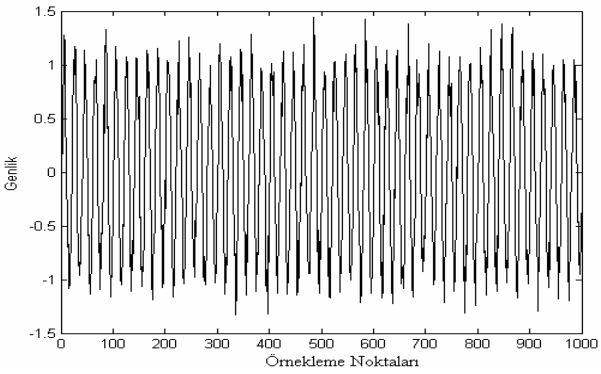
Algoritmanın performansını kontrol etmek amacıyla 30 kez 1000'er iterasyon koşturulmuş ve en iyi sonucu veren sonuç değerlendirilmiştir. 20. koşmada elde edilen ağırlıklara göre şekiller çizdirilmiştir.



*Şekil 3:* Bilgi işareti, s(n)

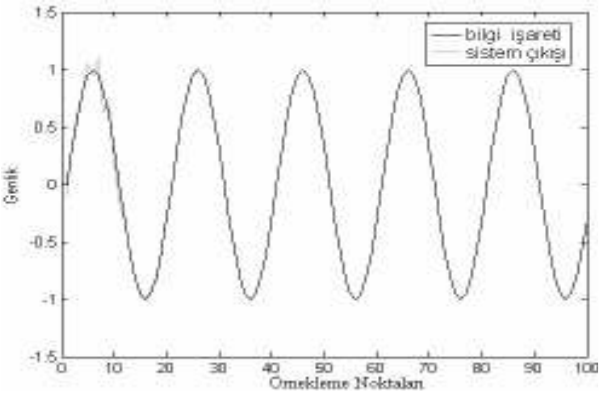



*Şekil 4:* Gürültü İşareti, $n_0(n)$

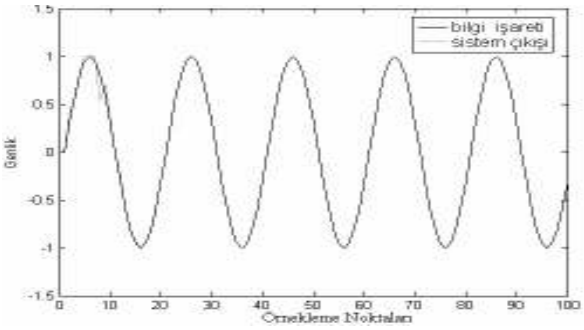



*Şekil 5:* Gürültülü işaret, $s(n)+n_0(n)$

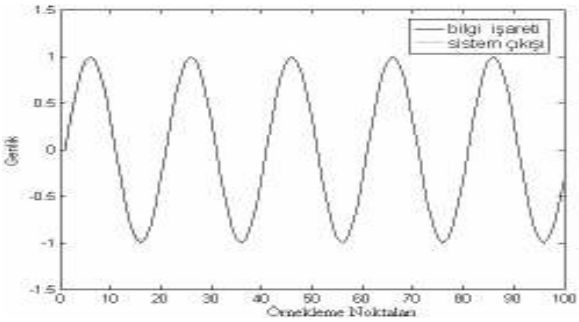
Şekil 3'de s(n) bilgi işareti gösterilmektedir. Şekil 4'de gürültü işareti ve Şekil 5'te de işaretin gürültüye maruz kalmış hali yani ana giriş olarak tanımlanan işaret, $d(n)=s(n)+n_0(n)$ verilmektedir.



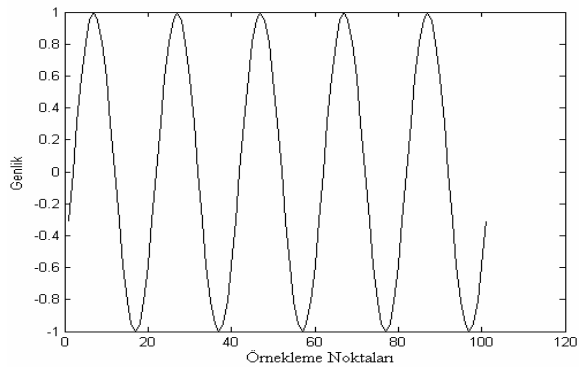

*(a)* SNR 8 dB de ilk 100 örnek




*(b)* SNR 10 dB de ilk 100 örnek




*(c)* SNR 13 dB de ilk 300 örnek




*(d)* SNR 8, 10, 13dB değerlerinde son 100 örnek

*Şekil 6:* SNR 8, 10, 13dB değerlerindeki süzgeç çıkışı

Şekil 6'da SNR 8, 10 ve 13 dB değerlerinde gürültü gidericiden elde edilen sonuçla istenilen bilgi işaretinin karşılaştırılması gösterilmektedir. Bütün SNR değerlerinde son 100 örnek için gürültü gidericinin çıkışında elde edilen simülasyon sonuçları aynıdır. Bunun için Şekil 6-(d) de bütün SNR değerleri için ortak olarak gösterilmiştir. Buradan da anlaşılmaktadır ki süzmede kullanılan hata ε, bu çalışma ile elde edilmek istenen işareti vermektedir. Artan SNR değerlerinde gürültü giderici istenildiği gibi gürültüyü çabuk ve başarılı bir şekilde çalışarak gidermektedir.

## 5. Sonuç

Bu çalışmada, FG algoritması kullanılarak adaptif gürültü giderici tasarlanmıştır. SNR değerleri 8, 10 ve

13dB olduğunda algoritma performansını değerlendirmek amacıyla 30 kez koşulmuş ve bu koşmaların 20. koşma sonucu elde edilen en iyi sonuç şekillerin çizdirilmesinde kullanılmıştır. Eğer adaptasyon algoritması olarak FG algoritması tercih edilirse, değişik SNR değerlerinde de başarılı bir adaptif gürültü giderici tasarlanabildiği simülasyon sonuçlarından görülmüştür. SNR değerlerini arttırdığımızda gürültü gidericinin daha kısa zamanda gürültüyü giderdiği gösterilmiştir.

## 6. Kaynakça


[1] Bernard Widrow and Samuel D.Stearns, Adaptive Signal Processing C:12, p:302-304,

[2] D.G.Mayer, B.P.Kingborn and A.A.Archer, "Differential Evolution an Easy and Efficient Evolutionary Algorithm For Model Optimization", Agricultural Systems, 2005.

[3] I.L.Lopez Cruz,L.G.Van Willigenburg and G.Van Straten, "Efficient Differential Evolution Algorithms for Multimodal Optimal Control Problems", Applied Soft Computing 3, p:97-122, 2003.

[4] Storn, R. and Price, K., "Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, V:11. Kluwer Academic Publishers, p:341 – 359, 1997.

[5] Derviş Karaboğa, Yapay Zeka Optimizasyon Algoritmaları, Nobel Yayınevi, İstanbul, 2004.

[6] Price K. V., "Differential Evolution: a Fast and Simple Numerical Optimizer", In: Smith, M., Lee, M., Keller, J., Yen., J. (eds.): Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS. IEEE Press, New York, p:524-527, 1996.

[7] Dervis Karaboga and Selcuk Okdem, "A Simple and Global Optimization Algorithm for Engineering Problems", Differential Evolution Algorithm. Turk J Elec Engin, Vol.12, No.1, 2004.

[8] Nurhan Karaboga, "Digital IIR Filter Design Using Differential Evolution Algorithm", EURASIP Applied Signal Processing, No.8, p:1269-1276, 2005.

[9] Price, K V., "An Introduction to Differential Evolution", In: Corne, D., Dorigo, M., and Glover, F. (eds): New Ideas in Optimization McGraw-Hill, London UK, p:79-108, 1999.