İSTANBUL TEKNİK ÜNİVERSİTESİ ★FEN EDEBİYAT FAKÜLTESİ
MATEMATİK MÜHENDİSLİĞİ PROGRAMI

PARÇACIK SÜRÜ OPTİMİZASYONU İLE MÜHENDİSLİK OPTİMİZASYON PROBLEMLERİ VE BAŞLANGIÇ DEĞER PROBLEMLERİNİN ÇÖZÜMÜ

BİTİRME ÖDEVİ

Damla UĞUR 090090447
Uğurcan KUŞKU 090080010

Danışman: Doç. Dr. Ali ERCENGİZ

MAYIS 2013

ÖNSÖZ

Bu çalışmayı hazırlamamızda bize yol gösteren, yardımını ve bilgisini hiçbir zaman esirgemeyen Sayın Hocamız Doç. Dr. Ali ERCENGİZ' e, öğrenim hayatımız boyunca her türlü desteği aldığımız değerli arkadaşlarımıza, bize sevgi, güven ve her türlü desteği veren ailelerimize en içten teşekkürlerimizi sunarız.

Mayıs, 2013

Damla UĞUR

Uğurcan KUŞKU

Anneme

İÇİNDEKİLER

ÖZET

Parçacık sürü optimizasyonu ilk olarak 1995 yılında Dr. Eberhart ve Dr. Kennedy tarafından geliştirilmiş popülasyon temelli sezgisel bir optimizasyon tekniğidir. Bu teknikte her bir parçacık problemi çözmede potansiyel bir adaydır.

Bu çalışmada popülasyon temelli olan parçacık sürü optimizasyonunun kökenleri, nasıl kullanıldığı, nerelerde kullanıldığı, nasıl geliştirildiği, problemlere nasıl uygulandığı gibi konulara yer verilmiştir.

Çalışmada; lineer ve nonlineer denklem sistemlerinin çözümü, mühendislik problemlerinin çözümü ve başlangıç değer problemlerinin çözümü işlenmiştir.

1. GİRİŞ

1.1. OPTİMİZASYON NEDİR?

Optimizasyon, bir problemin olası çözümler arasından en uygununu belirlemekle ilgilenen bir disiplindir. En uygun çözüm problem ve çözen kişiye bağlı olarak bir ya da bir kaç kritere bağlı olarak belirlenir. Örneğin yapısal bir mühendislik probleminde, çözüm temel mühendislikteki özelliklere bağlı olarak belirlenebileceği gibi dizayn eden kişinin estetik zevklerine de bağlı olabilir. Kısıtlamalar kullanıcıya ya da problemin kendisine bağlıdır. Eğer çözüm tüm kısıtlamaları karşılıyorsa uygun(optimal) çözümdür.

Optimizasyon problemlerinin değişik yapıları ve matematiksel karakteristikleri, nonlineerlik, süreklilik, dış bükeylik gibi ortak özellikleri olan özel problemlerin çözümünde kullanılacak algoritmalar geliştirilmesini gerektirmiştir. Günümüzde, çok çeşitli problem türlerinin çözümü için geliştirilmiş algoritmalar mevcuttur. Bununla beraber, aynı problem tipinin değişik örneklerinde kullanılmak üzere farklı hesaplama yöntemleri de geliştirilmiştir. Bu nedenle de optimizasyon teorisi ve uygulamaları için daha özel fikir ve yöntemlere olan ihtiyaç sürmektedir.

1.2. OPTİMİZASYON PROBLEMLERİNİN ÇEŞİTLERİ

Bir fonksiyon en genel şekliyle, $f: A \rightarrow Y$ olarak ifade edilir.(A: Tanım kümesi, Y: Değer kümesi olmak üzere.) Literatürde çoğu optimizasyon problemi, tanım kümesi n- boyutlu Euclid uzayı, R^n, ve değer kümesi reel sayıların bir kümesi olan fonksiyonların minimize edilmesini gerektirir.

Optimizasyon problemleri genel olarak amaç fonksiyonunun formuna göre çeşitlenir. Bunlardan en önemlileri aşağıda belirtildiği gibidir:

Lineer optimizasyon: Amaç fonksiyonu ve kısıtlar lineer olduğunda.
Non-lineer optimizasyon: Optimizasyon probleminde en az bir tane non-lineer fonksiyon olduğu durumları inceler.
Konveks optimizasyon: Amaç fonksiyonu ve uygun küme konveks olduğu durumları inceler.
Kuadratik optimizasyon: Amaç fonksiyonu kuadratik, kısıtlar lineer olduğu durumları inceler.
Stokastik optimizasyon: Kısıt ve parametrelerin rastgele değişkenlere bağlı olduğu durumları inceler.

1.3. OPTİMİZASYON ALGORİTMALARININ SINIFLANDIRILMASI

Optimizasyon problemleri genel olarak deterministik ve stokastik olarak iki ana kategoride ele alınır. Deterministik yaklaşım, bilinen parametreleri göz önüne alarak klasik yöntemlerle çözüm yöntemidir ve tasarımda oluşabilecek değişkenler(varyasyonlar) dikkate alınmaz. Bir varyasyon durumunda yaklaşım yetersiz kalacaktır. Stokastik yaklaşım ise değişkenlerin(varyasyonların) dikkate alındığı yöntemdir. Optimizasyon problemlerinin sınıflandırılmasında Garantili, direkt ve direkt olmayan olmak üzere çeşitli özel sınıflandırmalar kullanılması rağmen, tüm bu sınıflandırmalara bakılmaksızın, genel ve en etkili global optimizasyon metodu olan Evrimsel Hesaplama odaklanacağımız konu olacaktır.

1.3.1. EVRİMSEL HESAPLAMA

Evrimsel hesaplama optimizasyon problemlerinde kullanılan temel yöntemlerden biridir. Bu yöntemin algoritmaları, Darwin'in türlerin kökeni teorisinde geçen biyolojik prensiplere dayanmaktadır. Bu algoritmalar potansiyel çözüm popülasyonunu elde ederken iterasyonla evrimleşmeye dayanır. Evrim, popülasyondaki özel operatörlerin (çaprazlama, seleksiyon ve mutasyon gibi) çıktısı olarak kendini gösterir.

Darwin'in teorisinin optimizasyon alanındaki ilk uygulamaları 60'ların başında yapılmış olmasına rağmen evrimsel hesaplama olarak adlandırdığımız yöntem 90'larda kendini göstermiştir. Evrimsel hesaplamanın yaklaşımları olan genetik algoritmalar, evrimsel programlama, evrimsel strateji ve parçacık sürü optimizasyonu olarak bölümlere ayrılabilmektedir.

Yöntemleri daha iyi anlayabilmek adına, çoğu yöntemde operatör olan seleksiyon, çaprazlama ve mutasyon kavramlarının açıklaması aşağıda yer almaktadır.

1.3.1.1. EVRİMSEL HESAPLAMALARDA KULLANILAN EK BİLGİLER

Seleksiyon: Seleksiyon, bir popülasyondan, birbirleriyle çaprazlama yapacak bireylerin seçilme prosedürüdür. Birey seçimi deterministik ya da stokastik olabilmekle beraber her zaman için fonksiyon değerine bağlıdır. Deterministik yaklaşım, en küçük fonksiyon değerlerinde olan en iyi bireyleri seçerken, stokastik yaklaşım en yüksek ihtimalli en iyi bireyi atamaktadır.

Literatürde birçok seçilim şeması olmasına rağmen en genellerinden olan ***Tournament Selection*'ın** pseudocode'u aşağıdaki gibidir:

Do $(i = 1, \ldots, k)$

Choose randomly m individuals from the population P.

Select one among the m individuals.

Add the selected individual into the parents pool.

End Do

Çaprazlama: Çaprazlama, yeni bir döl üretme amacında olan iki bireyin genlerinde taşıdıkları bilgilerin yeniden yapılanması sürecidir. Yeni bireyin DNA'sı, bireyi oluşturanların DNA'larındaki genetik bilginin kombinasyonu şeklinde olacaktır. Buradaki amaç çeşitliliği arttırmaktır.

Mutasyon: Mutasyon temel bir biyolojik operasyon olup, organizmalara değişen çevre şartlarına daha uyumlu olmalarına olanak sağlamak için, genotiplerini değiştirerek bazı biyolojik özelliklerini değiştirme fırsatı sunar. Bu durum babadan oğula kalıtsal olarak geçebileceği gibi organizmanın kendisi tarafından sonradan da kazanılabilir.

Biyolojik olarak gerçekleşen bu durumun genetik algoritmalar için modellenmesi, popülasyondaki çeşitliliği desteklemekte, bireylerin birbirlerine benzemelerini engellemekte ve lokal minimumu durgunlaştırıp kısıtlayacak davranışlara engel olmaktadır.

Mutasyon aşağıdaki pseudocode'da belirtildiği gibi tanımlanmaktadır:

Do $\{i = 1, \ldots . N\}$

Do $(j = 1, \ldots . n)$

Generate a uniformly distributed random number, $r \in [0,1]$

If $(r < a)$ **Then Mutate** the component pij

End If

End Do

End Do

Yukarıdaki operatörlerin yardımını alan **Evrimsel Hesaplama Yöntemleri** aşağıda açıklanmaktadır.

1.3.1.2. GENETİK ALGORİTMALAR

Genetik algoritmalar, J.H. Holland tarafından 60'ların ortalarında geliştirilmiştir. Holland'ın o zamanlardaki araştırması, değişen çevre şartlarına adapte olup cevap verebilen sistemler olmuştu. Sürekli rekabet ve döllenme, uzun vadede, doğal sistemlerdeki adaptasyonun temelleri olarak görülüyordu. Bunların yapay adaptasyon sistemleri için modellenmesi fikri gayet cazibeli görünüyordu. Holland'ın bu fikri öğrencilerinin tezlerinde de genetik algoritmalar adında yeni bir optimizasyon tekniği olarak kendini göstermiştir. Bu teknik seleksiyon çaprazlama ve mutasyona dayanmakta olup bugünlerde kullanılan önemli optimizasyon yöntemlerinden biridir.

1.3.1.3. EVRİMSEL PROGRAMLAMA

Evrimsel programlama temel evrimsel hesaplama yöntemlerinden biri olup 60'ların ortalarında L.J. Fogel tarafından yapay zeka alanında kullanılmak üzere geliştirilmiştir. Yöntemdeki temel operatör ise mutasyondur.

1.3.1.4. EVRİMSEL STRATEJİ

Evrimsel strateji 60'larda Rechenberg, Schwefel ve Bienert tarafından mühendislik optimizasyon problemlerinin çözümü için geliştirilmiş bir optimizasyon tekniğidir. Bu teknikte sadece seleksiyon ve mutasyon kullanılmış olup çaprazlama yer almamaktadır.

2. PARÇACIK SÜRÜ OPTİMİZASYONU

2.1. KÖKENLERİ

Parçacık sürü optimizasyonunun temelleri Reynolds' un 1987 yılında yapmış olduğu kuş sürülerine benzer parçacıkların modellenmesine yönelik çalışmasına dayanır.

Bu teknikte gruptaki her bir üyenin davranışı, ortaya çıkan karmaşık yapıya rağmen, genlerinde saklı olan basit bilgiye dayanır.

Örneğin bir kuş sürüsünün uçuşu her bir kuşun hedefe ve anlık komşularına olan mesafesini koruma eğilimine göre basit bir şekilde, kabaca modellenebilir. Bu mesafe sürülerin boyutuna ve istenen davranışın ne olduğuna göre değişir. Örneğin; bir balık sürüsü tehlike altında değilken balıklar birbirlerine daha uzak mesafede hareket ederken, sürü yırtıcı bir hayvan tehdidiyle karşılaştığında daha küçük parçalara ayrılıp formunu değiştirerek kendini korumaya alır.

Bu modellemelerin optimizasyon problemlerinin çözümündeki potansiyellerini gördükten sonra Eberhart ve Kenneddy konu üzerine yoğunlaşmaya başlamışlardır. Çalışmalarının sonucunda ise 1995 yılında popülasyon temelli sezgisel bir optimizasyon tekniği olan PSO'yu geliştirmişlerdir.

2.2. KULLANILDIĞI YERLER

Bu optimizasyon tekniği Eberhart ve Kennedy'nin ardınan bir çok optimizasyon probleminin çözümünde de kullanılmıştır. Lineer ve lineer olmayan problemlerin çözümü, atış problemleri, çok bilinmeyenli denklemlerin köklerini bulma ve endüstriyel problemlerin çözümü bunlardan bazılarıdır.

Performans açısından bakıldığında diğer optimizasyon tekniklerine göre daha avantajlıdır.

2.3. TEKNİĞİN UYGULANIŞI

2.3.1. PROBLEM FORMÜLASYONU

Tüm optimizasyon problemlerinin temel amacı amaç fonksiyonu olan $f(\vec{x})$'i maksimum ya da minimum yapmaktır. x vektörü n boyutlu karar vektörü yada reel sayılardan oluşan bir karar değeridir. [2]

Parçacık sürü optimizasyonu sürü halinde hareket etme özelliği gösteren kuşların davranışlarından esinlenilerek geliştirilmiş ve çok geniş alanlardaki problemlerin çözümü için kullanılmaktadır. Temelde, popülasyondaki parçacıkların başlangıçta rastgele dağılmasına ve devam eden süreçteki içgüdüsel hareketlerine dayanmaktadır. Bu davranış belirli bir alanda bir amaca ulaşana kadar devam etmektedir.

Sürü içerisindeki kuşların içgüdüsel hareketlerine matematiksel bir pencereden baktığımızda durumu şu şekilde ifade edebiliriz.

Araştırma yapılacak uzay $A \subset R^n$ olsun ve $f: A \to Y \subseteq \mathbf{R}$ fonksiyonu tanımlansın. Tanıma olabildiğince sadık kalabilmek adına A tanım kümesinin her problem için uygun uzaydan alındığı varsayılmaktadır.

Daha öncede bahsedildiği gibi PSO popülasyon tabanlı bir optimizasyon metodudur. Bahsi geçen popülasyondan "sürü(S)" diye bahsedilmekte ve bu sürünün içerisindeki her bir bireye "parçacık(x_i)" denmektedir. "N" sürü içerisindeki toplam parçacık sayısını göstermekte, sürü ve parçacıklar ise şu şekilde tanımlanmaktadır:

$$S = \{x_1, x_2, \dots, x_N\} \quad (2.1)$$

Farklı sürüler içerisindeki parçacıklar değişiklik gösterebilmekte ve parçacıklar birden fazla parametreye bağlı olabilmektedirler. Parçacığın bağlı olduğu parametre sayısı ise "boyut" olarak adlandırılıp ve şu şekilde tanımlanmaktadır:

$$x_i = (x_{i1}, x_{i2}, \dots, x_{in})^T \in A, \quad i = 1,2, \dots, N \qquad (2.2)$$

Parçacıkların her bir iterasyonda gözlem yapılan uzayda(A) hareket ettikleri varsayılmaktadır. Bu durum ise hız ve pozisyon vektörlerinin ortam şartlarına uyum sağlamasıyla gerçekleşmektedir.

$$v_i = (v_{i1}, v_{i2}, \dots, v_{in})^T, \quad i = 1,2, \dots, N \qquad (2.3)$$

Hız vektörü, her adımda bir önceki adımdaki bilgilere göre güncellenmektedir. Parçacıklar her bir adımdaki en iyi pozisyonlarını akıllarında tutup, diğer hareketlerini bu en iyi pozisyonlara göre belirlemektedirler. Bu durumda, parçacıkların her bir iterasyon için bir en iyi pozisyon(pbest) değeri bulunmaktadır.

$$P = \{p_1, p_2, \dots, p_N\} \quad (2.4)$$

$$p_i = (p_{i1}, p_{i2}, \dots, p_{in})^T \in A, \quad i = 1,2, \dots, N \qquad (2.5)$$

Parçacıkların en iyi pozisyonları $argmin$ fonksiyonu kullanılarak belirlenmektedir. $Argmin$ fonksiyonu, bir optimizasyon probleminde amaç fonksiyonunun minimum değeri almasını sağlayacak olan argümanları veren fonksiyondur. En basit ifadesiyle minimumu veren değişken değeridir.

$$p_i(t) = arg_t min f_i(t) \qquad (2.6)$$

Sürü içerisindeki parçacıklar sezgisel olarak birbirlerini takip ettiklerinden, sürüdeki en iyi pozisyona sahip olan parçacık takip edilecektir. Bunun için pbestler arasından, $argmin$ fonksiyonu kullanılarak minimum değer seçilir ve bu değer global en iyi pozisyon (gbest) olarak adlandırılır.

$$p_g(t) = arg_i min f(p_i(t)) \quad (2.7)$$

Ebert & Kennedy 1995 yılındaki çalışmalarının ilk versiyonlarında aşağıdaki denklemlere ulaşmışlardır.

$$v_{ij}(t+1) = v_{ij}(t) + c_1 R_1 \left(p_{ij}(t) - x_{ij}(t)\right) + c_2 R_2 \left(p_{gj}(t) - x_{ij}(t)\right) \quad (2.8)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \qquad (2.9)$$

$$i = 1,2,\dots,N, \quad j = 1,2,\dots,n$$

2.3.2. PARÇACIKLARIN HAREKET STRATEJİLERİ

Parçacıkların hareket stratejileri sürü içindeki konumlarına(x), parçacıkların boyutlarına(n), hızlarına(v), ataletlerine(w), ve hız - konum denklemlerinin optimum sonucu vermesini sağlayan bazı katsayılara (c, K) bağlıdır.

2.3.2.1. PBEST- GBEST

İçgüdüsel olarak her bir parçacık 2 şekilde hareket etme eğilimi gösterir.

Birincisi, sürünün tamamı için en iyi pozisyon (**gbest)**'e yaklaşma,
İkincisi, kendi en iyi poziyonu (**pbest**)'i koruma eğilimidir.

Sürü içindeki parçacıkların pbest(personel best) değerleri şu şekilde belirlenir;

Başlangıçtaki pbest, rastgele atanmış başlangıç koşullarının değerlerine eşittir. Sonraki aşamalarda ise, bir önceki iterasyonda belirlenmiş olan pbest ile karşılaştırılarak yeni pbest değeri atanır. Eğer t. iterasyonadaki değer (t+1). iterasyondaki değerden büyük ise yeni pbest (t+1). iterasyondaki değerdir. Aksi taktirde, yeni pbest t. iterasyondaki değer olacaktır.

$$p_i(t+1) = \begin{cases} x_i(t+1), & eğer\ f\left(x_i(t+1)\right) \le f\left(p_i(t)\right), \\ p_i(t), & değilse \end{cases} \quad (2.10)$$

Sürünün gbest (global best) değeri ise şu şekilde belirlenir;

Her bir iterasyon için pbest değerlerinden $f(x)$ fonksiyonunu minimum yapan değer, gbest'tir. Her iterasyonda bu işlem tekrarlanarak gbest güncellenir.

$$p_g(t) = arg_i minf(p_i(t)) \quad (2.11)$$

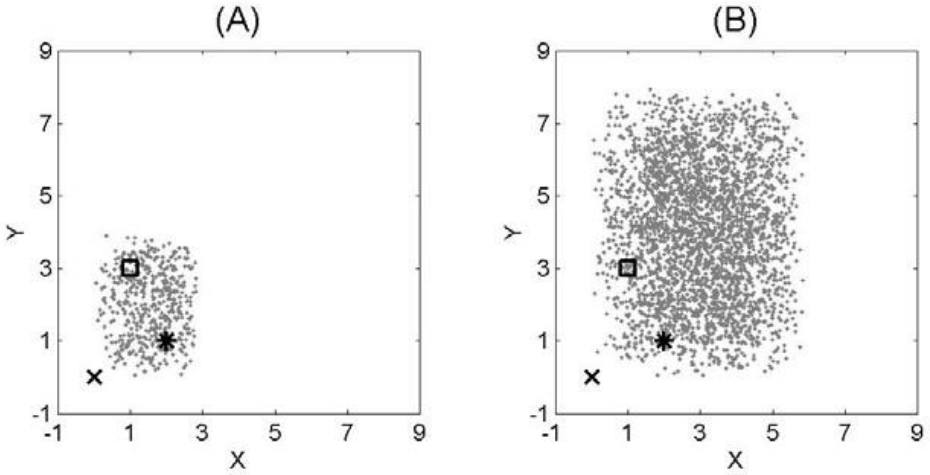
Sürü içerisindeki parçacıkların tüm boyutları için pbestler ve gbest ayrı ayrı hesaplanır.

2.3.2.2. R DEĞİŞKENİ

Parçacıkların boyutları optimizasyon sonuçlarını etkilemektedir. PSO tekniği doğadan esinlenilerek geliştirilmiş bir yöntem olduğu için parçacıkların boyutları göz ardı edilememektedir. Yapılan araştırmalara göre bu boyut değerlerinin [0,1] aralığında olması problemlerin çözümünde en iyi sonucu vermektedir.

2.3.2.3. C SABİTİ

Hız denkleminde yer alan C_1- C_2 değerleri parçacıkların sürü içerisindeki konumlarını etkilemektedir. C değerlerinin değişmesi aşağıdaki şekillerde görüldüğü gibi parçacıkların yayıldıkları alanı değiştirmektedir. Şekil A'da görüldüğü üzere $C_1=C_2=1.0$ alındığında parçacıklar daha dar alanda konumlanmışlardır. Şekil B'deki gibi $C_1=C_2=2$ alındığında ise daha geniş alana yayılmışlardır.



Yapılan araştırmalar $C_1=C_2=2$ değerinin en optimum sonucu verdiğini göstermektedir.

2.3.2.4. KONUM(X)

Sürü içerisindeki parçacıkların konumu başlangıçta rastgele belirlenmektedir. Rastgele üretilen bu başlangıç değerleri belirli sınır koşullarına bağlıdır. Sınır koşullarına bağlanmış olması sapmayı azaltmaktadır.

Parçacıkların konumu bir önceki iterasyondaki konumlarına ve o iterasyondaki hızlarına göre belirlenmektedir.

$$x_{ij}(t+1) = \begin{cases} a_j, & eğer\ x_{ij}(t+1) < a_{j,} \\ b_j, & eğer\ x_{ij}(t+1) < b_j \end{cases} \qquad (2.12)$$

$$i = 1,2,\dots,N, \qquad j = 1,2,\dots,n.$$

2.3.2.5. HIZ (V)

Sürü içerisindeki parçacıkların t anındaki hızları, bir sonraki iterasyondaki yeni konum ve yeni hızın belirlenmesinde etkilidir. (t+1). iterasyondaki V, t. iterasyondaki V ile (t+1). iterasyondaki X' in toplanmasıyla bulunur.

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \qquad (2.13)$$

$$i = 1,2,\dots,N.$$

Şimdiye kadar anlatılan bilgiler kullanılarak basit bazı problemler çözülebilmektedir. Parçacık sürü optimizasyonunun ana denklemleri kullanılarak lineer bir denklem sisteminin kökleri bulunmuştur. Çözülen denklem sistemi ve nasıl çözüldüğü uygulamalar kısmında "lineer denklem sistemlerinin köklerinin bulunması" altında anlatılmıştır.

2.3.3. YÖNTEMİN UYGULANIŞI

Her bir parçacık için, kısıtlara uygun bir şekilde başlangıç değerleri üretilir. Üretilen bu başlangıç değerleri ilk pbest' leri oluşturacaktır.

Parçacığın en iyi konumu (pbest) belirlendikten sonra daha önce anlatılan kurallara ve fonksiyonlara uygun bir şekilde bir gbest belirlenmelidir. Gbest hesabı komşuluk yönteminin kullanılıp kullanılmamasına göre değişiklik gösterir.

Atanan başlangıç değerlerine ve hesaplanan gbest ve pbest değerlerine göre konum vektörleri belirlenir.

Parçacıkların hız vektörleri belirlenir. Normalde bir önceki iterasyona göre şekillenecek olan bu vektör, ilk iterasyonda sıfır alınacaktır. Kuşların başlangıçta durdukları varsayılmaktadır.

Fonksiyon içerisinde kullanılmak üzere "c" ve "R" katsayıları belirlenmelidir. (Bu katsayıların nasıl belirleneceği yukarıda anlatılmıştır.) C_1 ve C_2 katsayıları her iterasyonda sabit kalırken R_1, R_2 katsayıları her iterasyonda ve iterasyon içerisindeki her bir işlemde rastgele belirlenmektedir.

Tüm değişkenler belirlendikten sonra daha önceden belirtilmiş olan hız ve konum denklerimde yerlerine yazılarak parçacığın iterasyon sonundaki hız ve konum değerlerine ulaşılmaktadır.

Bu işlemler belirli iterasyon sayısı sağlanana kadar ya da belirtilmiş olan bir kural gerçekleşene kadar devam ettirilirler. İşlemin tamamlanması durdurma kriterlerine bağlıdır. Durdurma kriterleri iterasyon sayısı veya en iyi sonuca yakınsama olarak belirlenebilir.

Yöntem uygulanırken sürü içerisindeki parçacıkların sürüden ayrılma veya sapma gibi ihtimallerine karşı herhangi bir önlem alınmamaktadır. Çeşitli sebeplerden dolayı yoğun denklem sistemleri ve kısıtları olan problemlerde bu durum problemin çözümünü zorlaştırmaktadır. Sapma durumlarının önüne geçebilmek için çeşitli önlemler almak gerekmektedir. Alınan bu önlemler ileride anlatılacaktır.

Girdiler:	N→ parçacık sayısı; S→ sürü; P → en iyi konum
Adım 1:	set t ← 0,
Adım 2:	**Initialize *S*** and **Set *P* ≡ *S***.
Adım 3:	**Evaluate *S*** and ***P***, and define index *g* of the best position.
Adım 4:	**While** (termination criterion not met)
Adım 5:	**Update *S*** using equations (1) and (2).
Adım 6:	**Evaluate *S*.**
Adım 7:	**Update *P*** and redefine index *g*.

Adım 8:	**Set** $t \leftarrow t+1$.
Adım 9:	**End While**
Adım 10:	**Print** best position found

Genel hatlarıyla parçacık sürü optimizasyon sisteminin nasıl çalıştığı "yöntemin uygulanışı" başlığı altında anlatılmıştır. Bu uygulama içerisinde yapılan işlemler parçacık sürü optimizasyonu yönteminin temelini oluşturmaktadır. Bu temel üzerine yeni eklemeler yapılarak optimizasyon problemlerinin ideale daha yakın sonuçlar vermesi sağlanabilmektedir. Bu eklemeler arasında başlıca atalet sabiti(W), K sabiti(K), hız kısıtı, konum kısıtı, penaltı(ceza) fonksiyonu ve komşuluk kavramı yer almaktadır. Bu kavramlar sırasıyla aşağıda anlatılmıştır.

2.3.3.1. ATALET SABİTİ (W)

Bazı çok bilinmeyenli ya da çok fonksiyonlu karmaşık optimizasyon problemlerinin çözümünde hız fonksiyonuna eklenen w, çözümün en ideal şekilde çıkmasına yardım eder.

$$w(t) = w_{up} - (w_{up} - w_{low})\frac{t}{T_{max}} \quad (2.14)$$

Atalet değişkeni hesaplanırken yukarıda belirtilmiş olan formül kullanılır. "t" değişkeni iterasyon sayısını göstermektedir. Denklemde görüldüğü üzere atalet değişkeni iterasyon satısı arttıkça artmaktadır.

2.3.3.2. K SABİTİ (K)

Çok karmaşık optimizasyon problemlerinde **W** kullanımı dahi, doğru sonucu vermede yeterli olmadığından **K** sabitine ihtiyaç duyulmuştur. Bu sabit, sapma sayısını azaltarak doğru sonuca ulaşılmasına olanak sağlamıştır. Yeni denklem aşağıda belirtildiği gibidir.

$$v_{ij}(t+1) = K\left[v_{ij}(t) + c_1 R_1\left(p_{ij}(t) - x_{ij}(t)\right) + c_2 R_2\left(p_{gj}(t) - x_{ij}(t)\right)\right], \quad (2.15)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2.16)$$

$$i = 1,2,\dots,N, \quad j = 1,2,\dots,n$$

Kappa sabiti belirlenirken C_1 ve C_2, 2'den büyük sayılar alınmalıdır.

$$\varphi = c_1 + c_2 > 4 \quad (2.17)$$

$$K = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad (2.18)$$

2.3.3.3. HIZ KISITI

Parçacıkların hızlarının büyüklükleri kontrolsüz bir şekilde arttığı taktirde sürü içerisinden kopmalar olacaktır. Böyle bir durumun gerçekleşmesi, PSO'nun zor problemlerde vereceği sonucu olumsuz bir şekilde etkilemektedir. Bu durumun önüne geçebilmek adına hız vektörüne sınırlamalar getirilmiştir.
İlk olarak tüm iterasyonlarda geçerli olacak olan V_{max} ve V_{min} değerleri aşağıdaki gibi belirlenir. Denklem içerisindeki "a" ve "b" değişkenleri konum kısıtının başlangıç ve bitiş değerleridir.

$$v_{max} = \frac{\min_i\{b_i - a_i\}}{k} \quad (2.19)$$

Sonraki iterasyonlardaki hız değeri, belirlenmiş olan V_{max} ve V_{min} değerleri arasında olmalıdır.

$$\left|v_{ij}(t+1)\right| = v_{max}, \quad i = 1,2,\dots,N \quad j = 1,2,\dots,n \quad (2.20)$$

$$v_{ij}(t+1) = \begin{cases} v_{max}, & eğer\ v_{ij}t+1() > v_{max} \\ -v_{max}, & eğer\ v_{ij}(t+1) < -v_{max} \end{cases} \quad (2.21)$$

2.3.3.4. KONUM KISITI

Başlangıçta her bir parçacık için konum kısıtlarına uygun değerler atanmaktadır ve ilk iterasyon bu değerler ve bu değerlere bağlı diğer değişkenlerle gerçekleştirilmektedir. Genel olarak yapılan uygulamada daha sonraki değerlerin bu kısıtlar içerisinde olup olmadıklarına bakılmamaktadır bu da bazı parçacıkların sürüden sapmalarının önüne geçilmemesine sebep olmuştur.

Parçacıkların sürü içerisinde kalıp optimizasyon problemini daha iyi bir çözüme kavuşturabilmek için her bir parçacığın yeni konumunun da belirli kısıtlar içinde olması gerekmektedir. Bu gerekliliği sağlayabilmek için konum kısıtı kullanılmış ve kısıt dışına çıkan parçacık yok sayılarak yerine yeni parçacık üretilmiştir.

Bu durum sapmayı tamamen ortadan kaldırarak ideal çözüme ulaşmada önemli bir adım olmuştur.

2.3.3.5. PENALTI (CEZA) FONKSİYONU

Son zamanlarda yapılan araştırmalarda kısıtları işlemek için penaltı fonksiyonu kullanılmıştır. Tek başına etkili olmayan bu ceza fonksiyonu yöntemi diğer kısıtlarla beraber kullanıldığında daha iyi sonuçlar elde edilmesine olanak sağlamıştır.

Penaltı fonksiyonunun ana mantığı kısıtların dışında kalan parçacıkların amaç fonksiyonunu büyüterek, problemin ideal çözüme ulaşmasına yardımcı olmaktır.

Algoritma içerisinde bu durum aşağıdaki gibi gerçekleşmektedir:

Kısıtlar kontrol edilir, eğer parçacık kısıt dışında ise algoritma içerisinde belirlenmiş bir değişkene (toplam) bu değer atanır.

Parçacığın her bir boyutu için bu işlem yinelenir. Bir sayaç değişkeni (sayac) yardımıyla bu işlemin kaç defa yapıldığı sayılır. Her parçacık için uygulanan bu fonksiyon (h) her iterasyonda yinelenir.

Algoritma sonunda geri döndürülecek fonksiyon, penaltı fonksiyonu algoritması içerisinde oluşturulan ve kullanılan değişkenleri içermektedir. Döndürülecek fonksiyon aşağıdaki şekilde oluşturulur:

Bir penaltı fonksiyonu oluşturulur.

$$h = 100 * sayaç + 100 * toplam \qquad (2.22)$$

Amaç fonksiyonu (f) ve penaltı fonksiyonu (h) toplanarak işleme girmek üzere geri döndürülür.

$$return(f + h) \qquad (2.23)$$

Penaltı fonksiyonu uygulandıktan sonra ortaya çıkan cevabı penaltı fonksiyonunun matematiksel yapısı çok fazla etkilemektedir. Eğer penaltı fonksiyonu azalarak artan bir fonksiyon ise sonuç ideale daha çok yaklaşacaktır.

$$h = 100000(sayac + toplam)^2 \qquad (2.24)$$

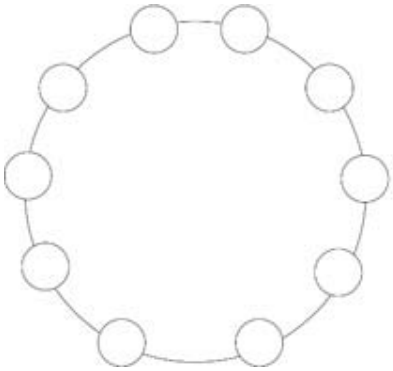
2.3.3.6. KOMŞULUK KAVRAMI

Parçacık sürü optimizasyonu algoritmalarında kavuşma kabiliyeti iyi olan parçacıklarla birlikte atalet değişkeni de işleme alınmıştı. Ancak bu durum daha kompleks problemlerde istenilen sonucu verememektedir.
Parçacıklardaki kayıp nedeniyle ilerleyen iterasyonlarda sürüde çöküş meydana gelmektedir. Bu çöküş incelemelerin devamını kısıtlamıştır ve parçacıkların sadece yerel pozisyonda iyi sonuç vermelerine sebep olmuştur. Parçacıkların hızlı kavuşması tek tepeli, dış bükey fonksiyonlu basit optimizasyon problemlerinde iyi sonuç vermesine rağmen; çok boyutlu, çok kısıtlı ve kompleks problemlerde büyük sorunlar yaşatmaktadır.

Karşılaşılan bu problemin üstesinden gelebilmek adına sürü içerisindeki tüm parçacıkların konumuna göre belirlenmiş olan gbest, her bir iterasyonda farklı bir şemaya göre belirlenmelidir.

Genel olarak, global bilgi şemasının yerine yerel bilgi şeması kullanılarak optimizasyon problemlerinin daha iyi sonuç vermeleri hedeflenmiştir.

Her bir parçacığın komşulukları bulunurken kuşların "halka topolojisine" uygun bir biçimde şekillendikleri düşünülmektedir.



Herhangi bir x parçacığının komşuluğu şu şekilde tanımlanmaktadır:

$$NB_i = \{x_{n_1}, x_{n_2}, \dots, x_{n_s}\}, \qquad \{n_1, n_2, \dots, n_s\} \subseteq \{1,2, \dots, N\} \qquad (2.25)$$

Bu komşuluk kümesinden amaç fonksiyonunu en küçük yapan değer o parçacık için yerel gbest' i ifade etmektedir.

$$p_{g_i} = \arg\min_{x_j \epsilon NB_i} f(p_j) \quad (2.26)$$

Parçacık sürü optimizasyonunun temel hız ve konum vektörü bu bilgilere göre güncellendiğinde aşağıdaki formüller çıkmaktadır:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1R_1\left(p_{ij}(t) - x_{ij}(t)\right) + c_2R_2(p_{g_ij}(t) - x_{ij}(t)) \quad (2.27)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2.28)$$

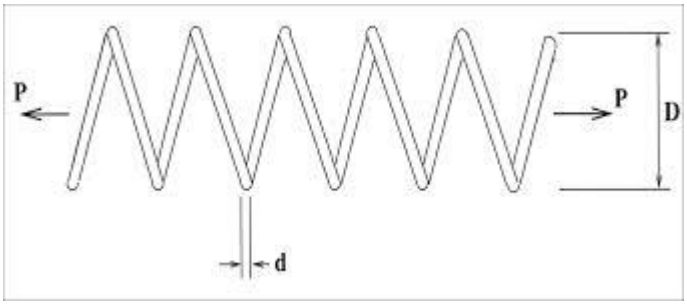
$i = 1,2,\dots,N, \quad j = 1,2,\dots,n.$

Şimdiye kadar yapılmış olan araştırmalar X_i parçacığının komşuluğunun [X_{i-3}, X_{i+3}] aralığında alınmasının optimal sonucu verdiğini göstermektedir.

Algoritmasına komşuluk özelliği eklenmeyen "hız düşürücü dizayn optimizasyon probleminde" optimal sonuca yaklaşım sağlanamamışken, bu bilgiler doğrultusunda algoritma güncellendiğinde problemin sonucu istenilen doğrultuda çıkmıştır.

3. UYGULAMALAR

3.1. YAY GERİLİM DİZAYNI OPTİMİZASYON PROBLEMİ



Şekil 3.1

Şekil 3.1 de yay gerilim dizaynı optimizasyon problemi gösterilmiştir. Bu problemin çözümünde parçacık sürü optimizasyonu yöntemi kullanılmış olup, optimal çözüme ulaşılmıştır. Yöntem 40 parçacık üzerinde uygulanmış ve C_1 ve C_2 sabitleri 2.7 alınmıştır.

Bu problemin minimize edilecek olan amaç fonksiyonu aşağıdaki gibidir:

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2$$

Kısıt fonksiyonları aşağıdaki gibidir:

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{7{,}178x_1^4} \leq 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12{,}566(x_2x_1^3) - x_1^4} + \frac{1}{5{,}108x_1^2} - 1 \leq 0$$

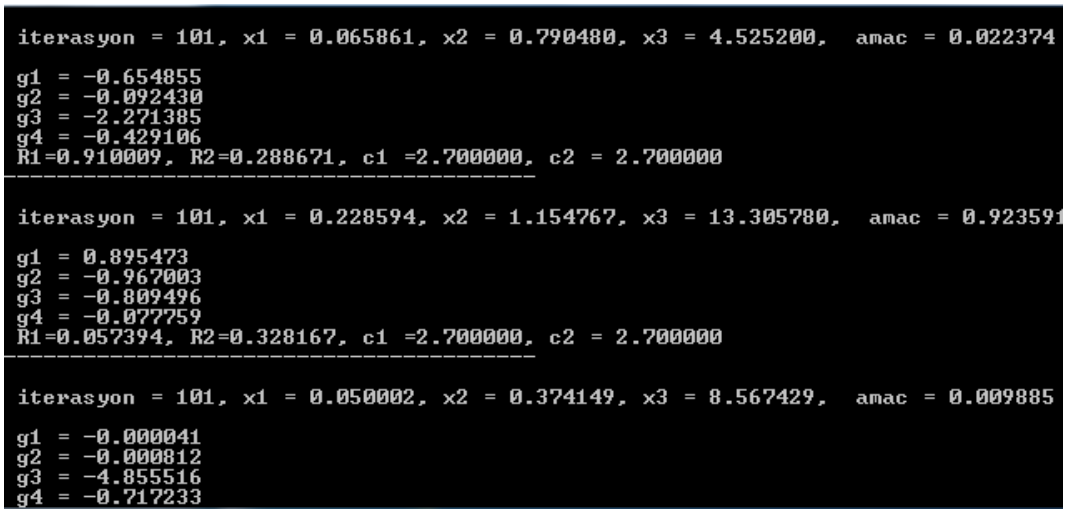
$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \leq 0$$

$$0.005 \leq x_1 \leq 2.0, \quad 0.25 \leq x_2 \leq 1.3, \quad 2.0 \leq x_3 \leq 15$$

Algoritmanın durdurulması için belirlenen kriter; daha iyi sonuca ulaşmaktır. İyi sonuç elde edildikçe sonuçlar ekrana yazılmaktadır. Problemin bilinmeyenlerinin değerleri ile kısıt fonksiyonlarının değerleri dosyaya yazılmaktadır.

Elde edilen sonuçlar referans alınan kaynaktaki sonuçlardan daha iyi çıkmıştır. Programın ekran çıktısı Şekil 3.2 de gösterilmiştir.



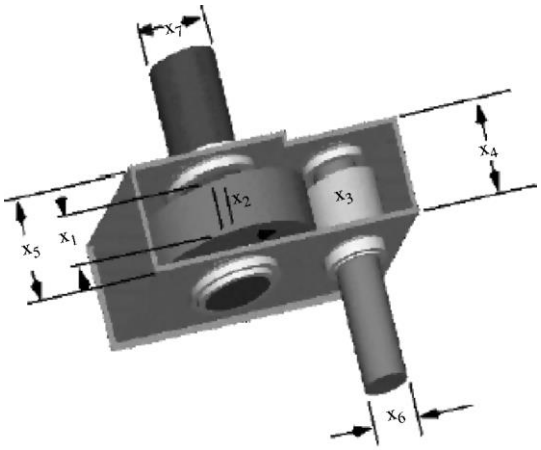
Şekil 3.2

Yay dizaynı optimizasyon probleminin parçacaık sürü optimizasyonu yoluyla çözümü ile referans alınan kaynaktaki çözümlerinin karşılaştırılması tablo 3.1 de gösterilmiştir.

	Bu çalışmadaki değerler	Referans alınan kaynaktaki değerler
x_1	0.050002	0.051583
x_2	0.374149	0.354190
x_3	8.567429	11.438675
$g_1(\vec{x})$	-0.000041	-2.000E-16
$g_2(\vec{x})$	-0.000812	-1.000E-16
$g_3(\vec{x})$	-4.855516	-4.048765
$g_4(\vec{x})$	-0.717233	-0.729483
$f(\vec{x})$	0.009885	0.012665

Tablo 3.1

3.2. HIZ DÜŞÜRÜCÜ OPTİMİZASYON PROBLEMİ



Şekil 3.3

Şekil 3.3 de hız düşürücü optimizasyon problemi gösterilmiştir. Bu problemin çözümünde parçacık sürü optimizasyonu yöntemi kullanılmış olup, optimal çözüme ulaşılmıştır. Yöntem 40 parçacık üzerinde uygulanmış ve C_1 ve C_2 sabitleri 2,8001 alınmıştır.

Bu optimizasyon probleminde minimize edilecek amaç fonksiyonu aşağıdaki gibidir:

$$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Kısıt fonksiyonları aşağıdaki gibidir:

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$$

$$g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^3} - 1 \leq 0$$

$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^3} - 1 \leq 0$$

$$g_5(\vec{x}) = \frac{1}{110x_6^3}\sqrt{(\frac{745x_4}{x_2x_3})^2 + 16.9 \times 10^6} - 1 \leq 0$$

$$g_6(\vec{x}) = \frac{1}{85x_7^3}\sqrt{(\frac{745x_5}{x_2x_3})^2 + 157.75 \times 10^6} - 1 \leq 0$$

$$g_7(\vec{x}) = \frac{x_2 x_3}{40} - 1 \leq 0$$

$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leq 0$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leq 0$$

$$g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0$$

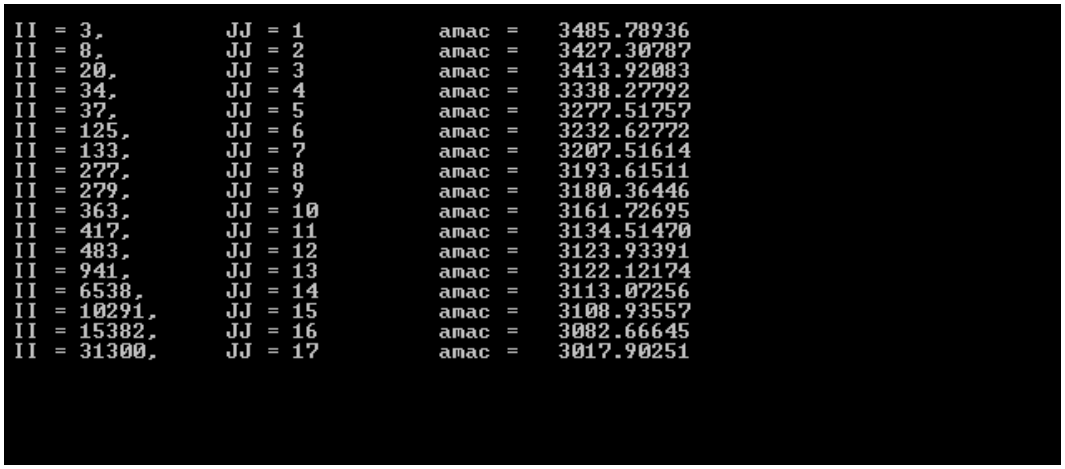
$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$

$$2.6 \leq x_1 \leq 3.6,\ \ 0.7 \leq x_2 \leq 0.8,\ \ 17 \leq x_3 \leq 28,\ \ 7.3 \leq x_4 \leq 8.3,\ \ 7.8 \leq x_5 \leq 8.3,$$

$$2.9 \leq x_6 \leq 3.9,\ \ 5.0 \leq x_7 \leq 5.5$$

Algoritmanın durdurulması için belirlenen kriter; daha iyi sonuca ulaşmaktır. İyi sonuç elde edildikçe sonuçlar ekrana yazılmaktadır. Problemin bilinmeyenlerinin değerleri ile kısıt fonksiyonlarının değerleri dosyaya yazılmaktadır.

Hız düşürücü optimizasyon probleminin parçacık sürü optimizasyonu metoduyla çözümünden elde edilen sonuçlar ile referans alınan kaynaktaki çözümler tablo 3.2 de karşılaştırılmıştır. Program çalıştırıldıktan sonra elde edilen ekran çıktısı şekil 3.4 te gösterilmiştir.



Şekil 3.4

	Bu çalışmadaki değerler	Referans alınan kaynaktaki değerler
x_1	3.503099	3.500000
x_2	0.700574	0.700000
x_3	17.016507	17
x_4	8.024138	7.30000
x_5	7.822384	7.800000
x_6	3.360747	3.350214
x_7	5.294963	5.286683
$g_1(\vec{x})$	-0.077145	-0.073915
$g_2(\vec{x})$	-0.201571	-0.197998
$g_3(\vec{x})$	-0.344330	-0.499172
$g_4(\vec{x})$	-0.901418	-0.901471
$g_5(\vec{x})$	-0.008140	0.000000
$g_6(\vec{x})$	-0.004682	-5.000E-16
$g_7(\vec{x})$	-0.004682	-0.702500
$g_8(\vec{x})$	-0.701967	-1.000E-16
$g_9(\vec{x})$	-0.583306	-0.583333
$g_{10}(\vec{x})$	-0.134970	-0.051325
$g_{11}(\vec{x})$	-0.012518	-0.010852
$f(\vec{x})$	3017.902510	2,996.348165

Tablo 3.2

3.3. DENKLEM SİSTEMLERİNİN KÖKLERİNİN BULUNMASI

$$f_1(\vec{x}) = 3 - x_1 x_3^2$$

$$f_2(\vec{x}) = x_3 \sin\left(\pi/x_2\right) - x_3 - x_4$$

$$f_3(\vec{x}) = -x_2 x_3 e^{(1-x_1 x_3)} + 0.2707$$

$$f_4(\vec{x}) = 2x_1^2 x_3 - x_2^4 x_3 - x_2$$

Parçacık sürü optimizasyon yöntemi lineer denklem sistemlerinin çözümünde de kullanılmaktadır. Yöntem yukarıdaki denklem sistemine uygulanmış olup, denklemlerin kökleri doğru bir şekilde bulunmuştur.

Uygulama 40 parçacık üzerinde yapılmış olup C_1 ve C_2 sabitleri 2.05 olarak alınmıştır. Parçacık sürü optimizasyonu ile denklem sistemlerinin kökleri

bulunurken mutlak değeri en küçük olan denklem seçilir ve işlemler bu denklem üzerinde yapılır.

Her iterasyonda denklemlerin değerleri tekrardan hesaplanır, mutlak değeri minimum olan fonksiyon değişiklik gösterebilir. Sıfıra en yakın olan yani mutlak değeri en küçük olan denklemde şartlar sağlandığında diğer denklemlerde de koşullar otomatikman sağlanacaktır.

Denklem sistemlerinin çözümünde belirli bir amaç fonksiyonu yoktur. Amaç fonksiyonu her iterasyonda değişebilir.

3.4. HIMMELBLAU NON-LİNEER OPTİMİZASYON PROBLEMİ

Parçacık sürü optimizasyon yöntemi non-lineer optimizasyon problemlerinin çözümünde de kullanılmaktadır. Yöntem bir mühendislik optimizasyon problemi olan Himmelblau optimizasyon probleminin çözümünde kullanılmıştır.

Uygulama 36 parçacık üzerinde yapılmış olup C_1 ve C_2 sabitleri 2.885 olarak alınmıştır.

3.5. BAŞLANGIÇ DEĞER PROBLEMLERİ

Bu çalışmada sürü optimizasyonu yöntemi ikinci mertebe lineer ve lineer olmayan başlangıç değer problemlerine uygulanmıştır. Yapılan bu uygulama ile ilgili literatürde herhangi bir çalışma bulunamamıştır. En genel ikinci mertebe başlangıç değer problemi;

$$L(y) = f(x),\ \ y(x_0) = \alpha,\ \ y'(x_0) = \beta,\ \ a < x$$

olarak ifade edilebilr. Burada L ikinci mertebe lineer yada lineer olmayan operatördür.

Sürü optimizasyonunu uygulamak için $x_i = x_0 + ih$, $(i = 1,2, ...)$ noktaları göz önüne alınarak, herhangi bir x_i için x_{i-1} de başlangıç koşullarını ve diferansiyel denklemi sağlayan çözümler

$y_i(x_m) = \sum_{k=0}^{m} a_k x^m$ şeklinde alınmıştır. Her x_i noktası için a_k katsayılarının her bir seçimi bir parçacık olarak göz önüne alınarak, x_i noktasında başlangıç koşullarını ve diferansiyel denklemi sağlayan en uygun a_k katsayıları belirlenmiştir.

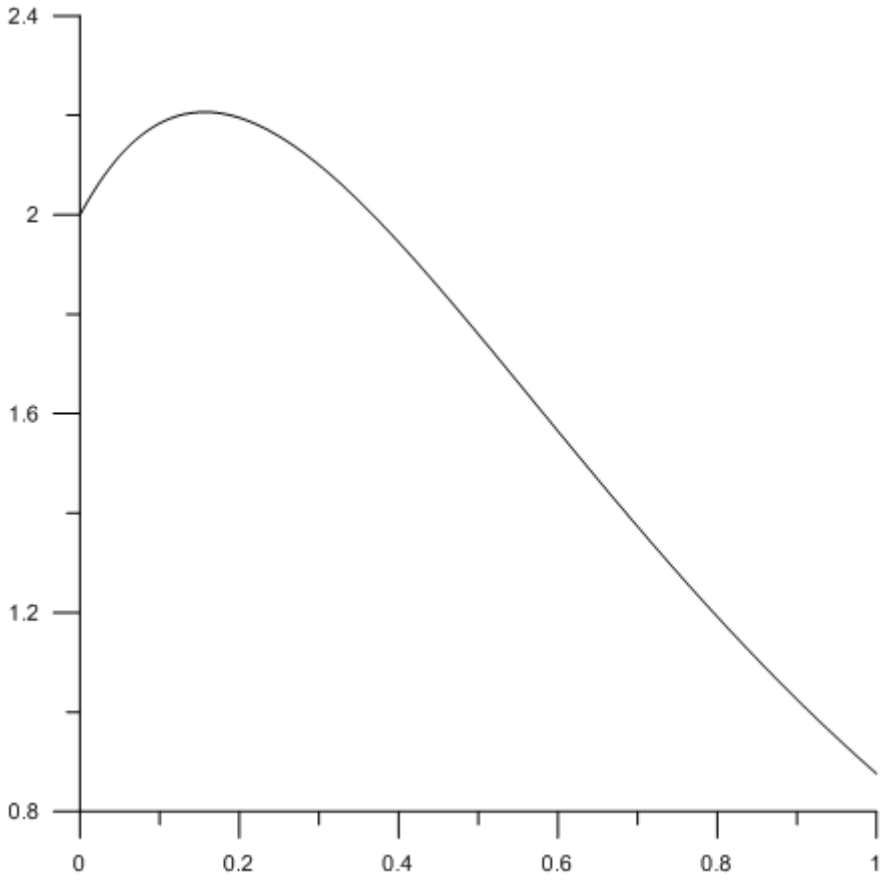
İzlenen yol şekil yardımıyla açıklanabilir. Önce $y_1(x_0) = \alpha,\ \ y_1'(x_0) = \beta$, $L(y_1) = f(x)$ başlangıç değer problemi $|y_1(x_0 - \alpha)|$, $|y_1'(x_0) - p|$ ve $\left|L\left(y_1(x_1)\right) - f(x_1)\right|$ fonksiyonları minimum (sıfır) olacak şekilde çözülüp, belirleniyor. $y_1(x_1)$ ve $y_1'(x_1)$ ardından $y_2(x_1) = y_1(x_1)$, $y_2'(x_1) = y_1'(x_1)$ve

$L(y_2) = f(x)$ başlangıç değer problemi çözülüyor. Herhangi i. adım için $y_i(x_{i-1}) = y_{i-n}(x_{i-1}),\ y_i'(x_{i-1}) = y_{i-1}'(x_{i-1}),\ L(y_i) = f(x)$ başlangıç değer problemi $|y_i(x_{i-1}) - y_{i-1}(x_{i-1})|$, $|y_i'(x_{i-1}) - y_i'(x_{i-1})|$ ve $\left|L\left(y_i(x_i)\right) - f(x_i)\right|$ ifadeleri minimum olacak şekilde çözülüyor. Aşağıda lineer ve lineer olmayan problemlere örnekler verilmiş ve tam çözüm (analitik çözüm) ile karşılaştırmaları yapılmıştır.

3.5.1. ÖRNEK 1

$y'' + 5y' + 6y = 0,\ \ y(0) = 2,\ \ y'(0) = 3$

Lineer başlangıç değer probleminin parçacık sürü optimizasyon çözümü ile analitik çözümü grafik 3.1 de gösterilmiştir. Analitik çözüm: $9e^{-2x} - 7e^{-3x}$.

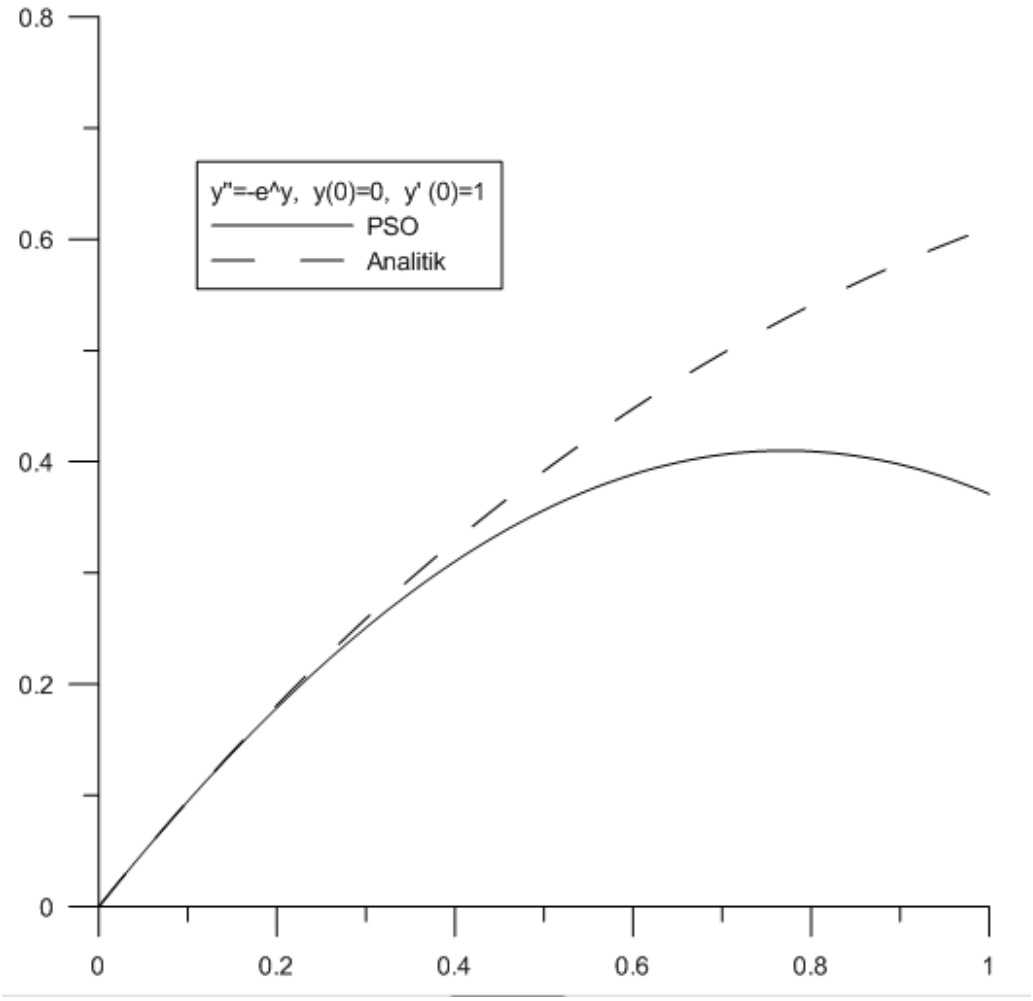


Grafik 3.1

3.5.2. ÖRNEK 2

$y'' = -e^y,\ y(0) = 0,\ y'(0) = 1$

Probleminin analitik çözümü $y(x) = \ln(1 + \sin(x))$ dir. Parçacık sürü optimizasyonu ve analitik çözümler grafik 3.2 de gösterilmiştir.

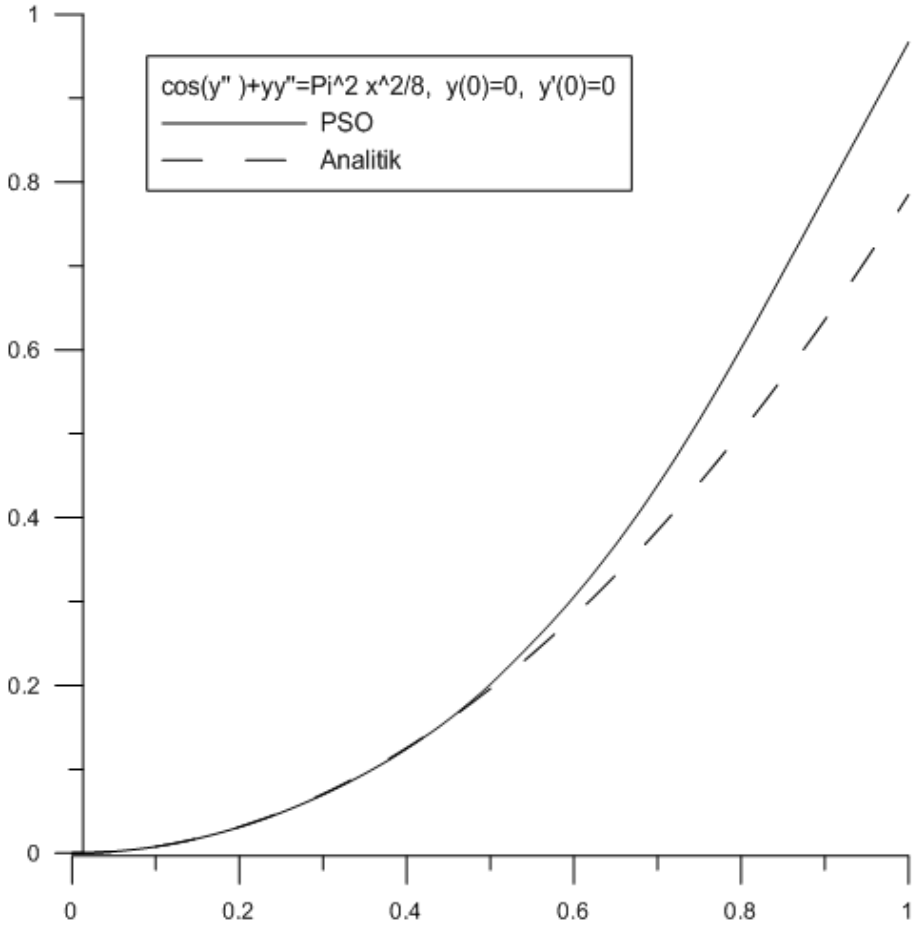


Grafik 3.2

3.5.3. ÖRNEK 3

$\cos(y'') + yy'' = \pi^2 x^2/8,\ y(0) = 0,\ y'(0) = 0$

Analitik çözüm: $y = \frac{\pi x^2}{4}$. Analitik çözüm ve sürü optimizasyonu çözümü grafik 3.3 te gösterilmiştir.



Grafik 3.3

3.5.4. ÖRNEK 4

İkinci mertebe lineer olmayan tekil problem göz önüne alınmıştır.

$$y'' + \frac{8y'}{x} + 18y = -4yln(y),\ y(0) = 1,\ y'(0) = 0$$

Analitik çözüm $y(x) = e^{-x^2}$ dir. Burada sonuçlar aşağıdaki tabloda gösterilmiştir.

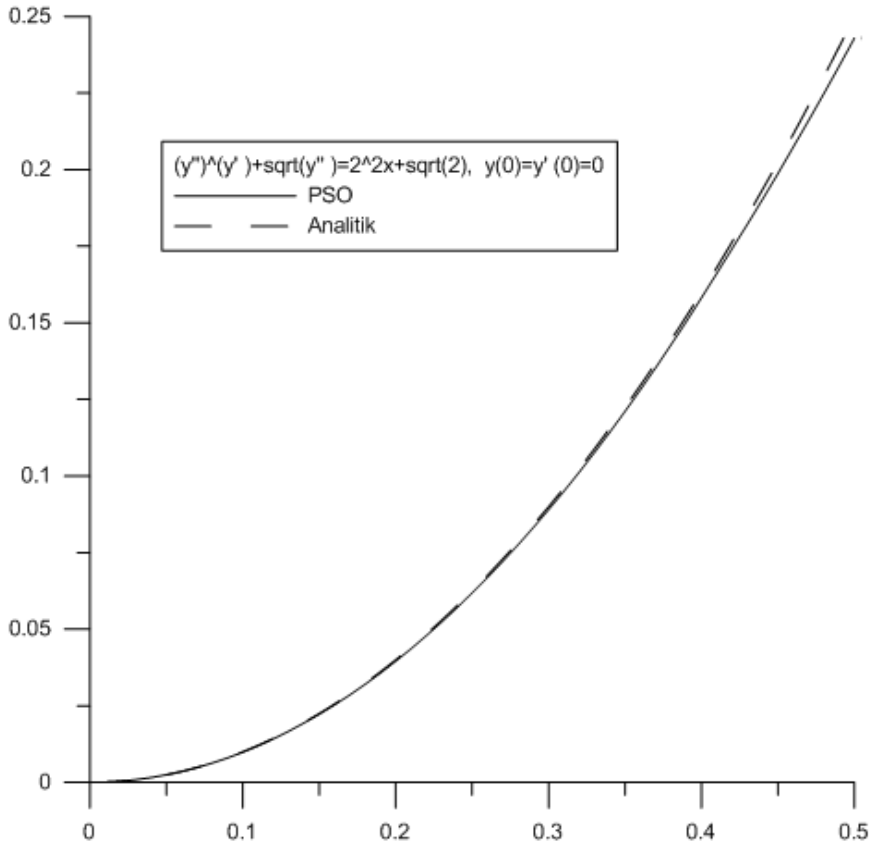
x	PSO	ANALİTİK
0.1	0.9900598	0.9900498
0.5	0.779266	0.7788007
0.8	0.5272338	0.5272924
1	0.367542	0.367879
2	0.0163824	0.0183156

Tablo 3.3

3.5.5. ÖRNEK 5

$$(y'')^{y'} + \sqrt{y''} = 2^{2x} + \sqrt{2},\ y(0) = y'(0) = 0$$

Analitik çözüm $y = x^2$. Sonuçlar grafik 3.4 te gösterilmiştir.



Grafik 3.4

Örneklerden görüleceği gibi lineer problemlerin çözümünde parçacık sürü optimizasyonu yöntemi çok daha iyi sonuç vermektedir. Lineer olmayan durumda x' in artan değerleri ile analitik çözümde olan fark büyümektedir. Bu çalışmada $y_i'(x)$ polinomlarının derecesi M=6 alınmıştır. Daha büyük M değerleri için, sözü edilen fark azaltılabilir.

4. EKLER

4.1. YAY GERİLİM DİZAYNI OPTİMİZASYON PROBLEMİ

File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

swarm11.c

```
1   #include <stdio.h>
2   #include <time.h>
3   #include<conio.h>
4   #include<math.h>
5   #define randomize()    srand((unsigned int)time(NULL))
6   #define frand()           (rand()%10000/10001.0)
7   /// STRING DESIGN OPTIMIZATION PROBLEM
8   typedef struct  nokta{
9           double x;
10          double y;
11          double z;
12
13          }NKT;
14  typedef struct parcacik{
15          NKT konum;
16          NKT pbest;
17          NKT hiz;
18          NKT gbest;
19          double amac;
20          }PCK;
21  #define c1   2.7
22  #define c2   2.7
23  #define N   40 // N parcacık sayısı
24  #define Pi  3.141592654
25  #define iter_say  100
26  PCK X[N];
27  double g1 ( double, double, double);
28  double g2 ( double, double, double);
29  double g3 ( double, double, double);
30  double g4 ( double, double, double);
```

```
31  main()
32  {
33     double R1, R2, min, K, F, vmax = .105, amacc, gg1,gg2,gg3,gg4;
34     PCK gb;
35     F = c1 + c2;
36     K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
37     int i, m, j, k, ii = 0;
38
39     PCK gbesti( int ); // ring topolojisini kullanarak r = 3 için i. parcacigin
40                        // gbest değerini buluyor
41    randomize();
42     double f( double, double, double);
43      double ff( double x1, double x2, double x3);
44   do
45    {
46
47    for ( i = 0; i< N; i++)
48
49       {
50         X[i].konum.x = X[i].pbest.x = (2-0.05)*frand()+0.05;
51         X[i].konum.y = X[i].pbest.y = (1.3-0.25)*frand()+0.25;
52         X[i].konum.z = X[i].pbest.z = (15-2)*frand()+2;
53         X[i].amac = f( X[i].konum.x, X[i].konum.y, X[i].konum.z);
54         X[i].hiz.x = 0;
55         X[i].hiz.y = 0;
56         X[i].hiz.z = 0;
57
58       }
59
60     R1 = frand();
```


```
31  main()
32  {
33     double R1, R2, min, K, F, vmax = .105, amacc, gg1,gg2,gg3,gg4;
34     PCK gb;
35     F = c1 + c2;
36     K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
37     int i, m, j, k, ii = 0;
38
39     PCK gbesti( int ); // ring topolojisini kullanarak r = 3 için i. parcacigin
40                        // gbest değerini buluyor
41    randomize();
42     double f( double, double, double);
43      double ff( double x1, double x2, double x3);
44   do
45    {
46
47    for ( i = 0; i< N; i++)
48
49       {
50         X[i].konum.x = X[i].pbest.x = (2-0.05)*frand()+0.05;
51         X[i].konum.y = X[i].pbest.y = (1.3-0.25)*frand()+0.25;
52         X[i].konum.z = X[i].pbest.z = (15-2)*frand()+2;
53         X[i].amac = f( X[i].konum.x, X[i].konum.y, X[i].konum.z);
54         X[i].hiz.x = 0;
55         X[i].hiz.y = 0;
56         X[i].hiz.z = 0;
57
58       }
59
60     R1 = frand();
```

```
61      R2 = frand();
62      printf("\n R1=%f, R2=%f, c1 =%f, c2 = %f", R1, R2, c1, c2);
63
64
65      for(i = 0; i < N; i++)
66        {
67            gb = gbesti(i);
68            X[i].gbest.x = gb.pbest.x;
69            X[i].gbest.y = gb.pbest.y;
70            X[i].gbest.z = gb.pbest.z;
71
72         }
73      for( j = 1; j<= iter_say; j++)
74      {
75        R1 = frand();
76        R2 = frand();
77
78       for( i = 0; i < N; i++)
79        {
80         X[i].hiz.x = K*(X[i].hiz.x + c1*R1* ( X[i].pbest.x - X[i].konum.x)
81                     + c2*R2*(X[i].gbest.x -  X[i].konum.x ));
82         if( X[i].hiz.x > vmax )  X[i].hiz.x = vmax;
83         if( X[i].hiz.x < -vmax ) X[i].hiz.x = -vmax;
84
85         X[i].hiz.y = K*(X[i].hiz.y + c1*R1* ( X[i].pbest.y - X[i].konum.y)
86                     + c2*R2*(X[i].gbest.y -  X[i].konum.y ));
87
88         if( X[i].hiz.y > vmax )  X[i].hiz.y = vmax;
89         if( X[i].hiz.y < -vmax ) X[i].hiz.y = -vmax;
90
```

```
91          X[i].hiz.z = K*(X[i].hiz.z + c1*R1* ( X[i].pbest.z - X[i].konum.z)
92                      + c2*R2*(X[i].gbest.z -  X[i].konum.z ));
93
94           if( X[i].hiz.z > vmax )  X[i].hiz.z = vmax;
95           if( X[i].hiz.z < -vmax ) X[i].hiz.z = -vmax;
96
97          X[i].konum.x = X[i].konum.x + X[i].hiz.x;
98          X[i].konum.y = X[i].konum.y + X[i].hiz.y;
99          X[i].konum.z = X[i].konum.z + X[i].hiz.z;
100
101
102         if( X[i].konum.x < 0.05 )
103           {
104            X[i].konum.x = (2-0.05)*frand()+0.05;
105            X[i].konum.y = (1.3-0.25)*frand()+0.25;
106            X[i].konum.z = (15-2)*frand()+2;
107          }
108
109         if( X[i].konum.x > 2 )
110          {
111           X[i].konum.x = (2-0.05)*frand()+0.05;
112           X[i].konum.y = (1.3-0.25)*frand()+0.25;
113           X[i].konum.z = (15-2)*frand()+2;
114          }
115         if( X[i].konum.y < 0.25 )
116          {
117           X[i].konum.y = (1.3-0.25)*frand()+0.25;
118           X[i].konum.x = (2-0.05)*frand()+0.05;
119           X[i].konum.z = (15-2)*frand()+2;
120           }
```

```
121         if( X[i].konum.y > 1.3 )
122          {
123            X[i].konum.y = (1.3-0.25)*frand()+0.25;
124            X[i].konum.x = (2-0.05)*frand()+0.05;
125            X[i].konum.z = (15-2)*frand()+2;
126           }
127
128
129
130         if( X[i].konum.z < 2)
131          {
132            X[i].konum.y = (1.3-0.25)*frand()+0.25;
133            X[i].konum.x = (2-0.05)*frand()+0.05;
134            X[i].konum.z = (15-2)*frand()+2;
135          }
136         if( X[i].konum.z > 15 )
137          {
138            X[i].konum.y = (1.3-0.25)*frand()+0.25;
139            X[i].konum.x = (2-0.05)*frand()+0.05;
140            X[i].konum.z = (15-2)*frand()+2;
141          }
142
143
144         if( f( X[i].konum.x, X[i].konum.y,X[i].konum.z ) <
145             f( X[i].pbest.x, X[i].pbest.y,X[i].pbest.z ))
146            {
147                 X[i].pbest.x = X[i].konum.x;
148                 X[i].pbest.y = X[i].konum.y;
149                 X[i].pbest.z = X[i].konum.z;
150
```

```
151             }
152
153       }
154
155        for( i = 0; i < N; i++)
156           X[i].amac = f( X[i].pbest.x, X[i].pbest.y, X[i].pbest.z);
157
158        for( i = 0; i < N; i++)
159        {     gb = gbesti(i);
160              X[i].gbest.x = gb.pbest.x;
161              X[i].gbest.y = gb.pbest.y;
162              X[i].gbest.z = gb.pbest.z;
163
164        }
165
166    }
167
168   min = X[0].amac;
169      m =0;
170      for(k = 1; k < N; k++)
171    {
172       if( X[k].amac < min )
173        {
174          min = X[k].amac;
175
176          m = k;
177
178        }
179
180     }
```

```
amacc = ff( X[m].gbest.x, X[m].gbest.y,X[m].gbest.z);
 printf("\n--------------------------------------\n");
  printf("\n iterasyon = %d, x1 = %f, x2 = %f, x3 = %f,  amac = %f ", j,
  X[m].gbest.x, X[m].gbest.y,

   X[m].gbest.z, amacc);
   int printf (const char *__format, ...)
 gg1 = g1( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z);
 printf("\n g1 = %f",gg1 );
 gg2 = g2( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z);
 printf("\n g2 = %f", gg2);
 gg3 = g3( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z);
 printf("\n g3 = %f", gg3);
 gg4 = g4( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z);
 printf("\n g4 = %f", gg4);

 // getch();
 }while( gg1>0 || gg2>0 || gg3>0 || gg4>0|| amacc>0.010);
getch();
}


////////////////////////////
double f( double x1, double x2, double x3)
{
     double f1, h, g11, g22, g33, g44;
     double sayac = 0, top = 0;


     f1 = (x3+2)*x2*x1*x1;
```

```

     g11 = g1 ( x1, x2, x3 );
     g22 = g2 ( x1, x2, x3 );
     g33 = g3 ( x1, x2, x3 );
     g44 = g4 ( x1, x2, x3 );
     if( g11 > 0)
      {
        sayac++;
        top = top +g11;
       }
     if( g22>0)
      {
        sayac++;
        top = top +g22;
       }
     if( g33>0)
      {
        sayac++;
        top = top +g33;
       }
     if( g44>0)
      {
        sayac++;
        top = top + g44;
       }
      h = 100.*sayac+ 100.*top;
      return( f1 + h );


}
```

```
241
242   ////////////////////////////////
243
244   PCK gbesti ( int i)
245   {
246    PCK temp[11];
247    int k, j, jj, kk = 0, m;
248    double min;
249     k = i;
250
251    for( jj =1; jj<=5; jj++)
252    {
253         j = ( N + ( ++ i))% (N);
254         temp[kk++] = X[j];
255         j = ( N + ( -- k))% (N);
256         temp[kk++]= X[j];
257
258     }
259         temp[kk]=X[i];
260      min = temp[0].amac;
261      m = 0;
262      for( jj = 1; jj< kk; jj++ )
263      {
264         if( min > temp[jj].amac )
265          {min = temp[jj].amac; m  = jj; }
266       }
267   return(temp[m]);
268   }
269   ////////////////////////////////
270   double g1 ( double x1, double x2, double x3 )
```

```
271   {
272       return( 1-x2*x2*x2*x3/(71785*x1*x1*x1*x1));
273   }
274   ////////////////////
275   double g2 ( double x1, double x2, double x3 )
276   {
277      return( (4*x2*x2-x1*x2)/(12566*x2*x1*x1*x1-x1*x1*x1*x1)+1./(5108*x1*x1)-1);
278   }
279   ////////////////////
280   double g3 ( double x1, double x2, double x3 )
281   {
282      return(1-140.45*x1/(x2*x2*x3) );
283   }
284   ////////////////////
285   double g4 ( double x1, double x2, double x3 )
286   {
287      return((x2+x1)/1.5-1 );
288   }
289   ////////////////////
290   double ff( double x1, double x2, double x3)
291   {
292        double f1;
293
294        f1 = (x3+2)*x2*x1*x1;
295        return( f1);
296   }
```

Compiler | Resources | Compile Log | Debug | Find Results

ine: 184 Col: 3 Sel: 0 Lines: 296 Length: 7640 Insert Modified

```
iterasyon = 101, x1 = 0.065861, x2 = 0.790480, x3 = 4.525200,  amac = 0.022374

g1 = -0.654855
g2 = -0.092430
g3 = -2.271385
g4 = -0.429106
R1=0.910009, R2=0.288671, c1 =2.700000, c2 = 2.700000
----------------------------------------

iterasyon = 101, x1 = 0.228594, x2 = 1.154767, x3 = 13.305780,  amac = 0.923591

g1 = 0.895473
g2 = -0.967003
g3 = -0.809496
g4 = -0.077759
R1=0.057394, R2=0.328167, c1 =2.700000, c2 = 2.700000
----------------------------------------

iterasyon = 101, x1 = 0.050002, x2 = 0.374149, x3 = 8.567429,  amac = 0.009885

g1 = -0.000041
g2 = -0.000812
g3 = -4.855516
g4 = -0.717233
```

4.2. HIZ DÜŞÜRÜCÜ OPTİMİZASYON PROBLEMİ

File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

[*] swarm14.c

```
#include <stdio.h>
#include <time.h>
#include<conio.h>
#include<math.h>
#define randomize()    srand((unsigned int)time(NULL))
#define frand()           (rand()%10000/10001.0)

// speed reducer design optimization problem
typedef struct  nokta{
        double x;
        double y;
        double z;
        double u;
        double v;
        double a;
        double b;
        }NKT;
typedef struct parcacik{
        NKT konum;
        NKT pbest;
        NKT hiz;
        NKT gbest;
        double amac;
        }PCK;
#define c1   2.8001
#define c2   2.8001
#define N   40// N parcacık sayısı
#define Pi  3.141592654
#define iter_say  100

PCK X[N];
double g1 ( double, double, double, double, double, double, double );
double g2 ( double, double, double, double, double, double, double );
double g3 ( double, double, double, double, double, double, double );
double g4 ( double, double, double, double, double, double, double );
double g5 ( double, double, double, double, double, double, double );
double g6 ( double, double, double, double, double, double, double );
double g7 ( double, double, double, double, double, double, double );
double g8 ( double, double, double, double, double, double, double );
double g9 ( double, double, double, double, double, double, double );
double g10 ( double, double, double, double, double, double, double );
double g11 ( double, double, double, double, double, double, double );

main()
{
   FILE  *g;
   double R1, R2, min, K, F, vmax , amacc, gg1,gg2,gg3,gg4, amacc1 = 300000;
   double gg5, gg6, gg7, gg8, gg9, gg10, gg11;
   PCK gb;
   double a1, b1, a2, b2, a3, b3, a4, b4, a5, b5, a6, b6, a7, b7, vmaxx, vmaxy,
   g = fopen ("dosya1.dat", "w");
   int ii =1;

   F = c1 + c2;
   K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
   int i, m, j, k, jj = 0;
   a1 = 2.6;
   b1 = 3.6;

   a2 = 0.7;
```

```
61      b2 = 0.8;
62
63      a3 = 17.;
64      b3 = 28.;
65
66      a4 = 7.3;
67      b4 = 8.3;
68
69      a5 = 7.8;
70      b5 = 8.3;
71
72      a6 = 2.9;
73      b6 = 3.9;
74
75      a7 = 5.;
76      b7 = 5.5;
77
78      double find_min(int);
79      double find_max( int );
80      PCK gbesti( int ); // ring topolojisini kullanarak r = 3 için i. parcacigin
81
82                              // gbest değerini buluyor
83    randomize();
84     double f( int, double, double, double, double, double, double, double);
85      double ff( double x1, double x2, double x3, double x4, double x5,
86      double x6, double x7);
87    /*vmaxx = ( b1- a1)/10;
88    vmaxy = ( b2 - a2 )/10;
89    vmaxz = ( b3 - a3 )/10;
90    vmaxu = ( b4 - a4 )/10;*/
```

```
91      vmax = ( b1 - a1 );
92      if( vmax > (b2 - a2) ) vmax = b2 - a2;
93      if( vmax > (b3 - a3) ) vmax = b3 - a3;
94      if( vmax > (b4 - a4) ) vmax = b4 - a4;
95      if( vmax > (b5 - a5) ) vmax = b5 - a5;
96      if( vmax > (b6 - a6) ) vmax = b6 - a6;
97      if( vmax > (b7 - a7) ) vmax = b7 - a7;
98      vmax = vmax/2;
99    do
100    {
101
102   for ( i = 0; i< N; i++)
103
104      {
105        X[i].konum.x = X[i].pbest.x = (b1 - a1)*frand()+ a1;
106        X[i].konum.y = X[i].pbest.y = (b2 - a2)*frand()+ a2;
107        X[i].konum.z = X[i].pbest.z = (b3 - a3)*frand()+ a3;
108        X[i].konum.u = X[i].pbest.u = (b4 - a4)*frand()+ a4;
109        X[i].konum.v = X[i].pbest.v = (b5 - a5)*frand()+ a5;
110        X[i].konum.a = X[i].pbest.a = (b6 - a6)*frand()+ a6;
111        X[i].konum.b = X[i].pbest.b = (b7 - a7)*frand()+ a7;
112
113        X[i].amac = f(2, X[i].konum.x, X[i].konum.y, X[i].konum.z,
114        X[i].konum.u,X[i].konum.v, X[i].konum.a, X[i].konum.b);
115        X[i].hiz.x = 0;
116        X[i].hiz.y = 0;
117        X[i].hiz.z = 0;
118        X[i].hiz.u = 0;
119        X[i].hiz.v = 0;
120        X[i].hiz.a = 0;
```

```
121       X[i].hiz.b = 0;
122     }
123
124
125   R1 = frand();
126   R2 = frand();
127
128
129
130   for(i = 0; i < N; i++)
131     {
132         gb = gbesti(i);
133         X[i].gbest.x = gb.pbest.x;
134         X[i].gbest.y = gb.pbest.y;
135         X[i].gbest.z = gb.pbest.z;
136         X[i].gbest.u = gb.pbest.u;
137         X[i].gbest.v = gb.pbest.v;
138         X[i].gbest.a = gb.pbest.a;
139         X[i].gbest.b = gb.pbest.b;
140      }
141   for( j = 1; j<= iter_say; j++)
142   {
143
144
145
146     R1 = frand();
147     R2 = frand();
148
149     for( i = 0; i < N; i++)
```

```
150     {
151       X[i].hiz.x = K*(X[i].hiz.x + c1*R1* ( X[i].pbest.x - X[i].konum.x)
152                   + c2*R2*(X[i].gbest.x -  X[i].konum.x ));
153       if( X[i].hiz.x > vmax )  X[i].hiz.x = vmax;
154       if( X[i].hiz.x < -vmax ) X[i].hiz.x = -vmax;
155
156       X[i].hiz.y = K*(X[i].hiz.y + c1*R1* ( X[i].pbest.y - X[i].konum.y)
157                   + c2*R2*(X[i].gbest.y -  X[i].konum.y ));
158
159       if( X[i].hiz.y > vmax )  X[i].hiz.y = vmax;
160       if( X[i].hiz.y < -vmax ) X[i].hiz.y = -vmax;
161
162       X[i].hiz.z = K*(X[i].hiz.z + c1*R1* ( X[i].pbest.z - X[i].konum.z)
163                   + c2*R2*(X[i].gbest.z -  X[i].konum.z ));
164
165       if( X[i].hiz.z > vmax )  X[i].hiz.z =vmax;
166       if( X[i].hiz.z < -vmax ) X[i].hiz.z =-vmax;
167
168       X[i].hiz.u = K*(X[i].hiz.u + c1*R1* ( X[i].pbest.u - X[i].konum.u)
169                   + c2*R2*(X[i].gbest.u -  X[i].konum.u ));
170
171       if( X[i].hiz.u > vmax )  X[i].hiz.u =  vmax;
172       if( X[i].hiz.u < -vmax ) X[i].hiz.u = -vmax;
173
174       X[i].hiz.v = K*(X[i].hiz.v + c1*R1* ( X[i].pbest.v - X[i].konum.v)
175                   + c2*R2*(X[i].gbest.v -  X[i].konum.v ));
176
177       if( X[i].hiz.v > vmax )  X[i].hiz.v = vmax;
178       if( X[i].hiz.v < -vmax ) X[i].hiz.v = -vmax;
179
```

```
180          X[i].hiz.a = K*(X[i].hiz.a + c1*R1* ( X[i].pbest.a - X[i].konum.a)
181                      + c2*R2*(X[i].gbest.a -  X[i].konum.a ));
182
183          if( X[i].hiz.a > vmax )  X[i].hiz.a =vmax;
184          if( X[i].hiz.a < -vmax ) X[i].hiz.a =-vmax;
185
186          X[i].hiz.b = K*(X[i].hiz.b + c1*R1* ( X[i].pbest.b - X[i].konum.b)
187                      + c2*R2*(X[i].gbest.b -  X[i].konum.b ));
188
189          if( X[i].hiz.b > vmax )  X[i].hiz.b =  vmax;
190          if( X[i].hiz.b < -vmax ) X[i].hiz.b = -vmax;
191
192
193          X[i].konum.x = X[i].konum.x + X[i].hiz.x;
194          X[i].konum.y = X[i].konum.y + X[i].hiz.y;
195          X[i].konum.z = X[i].konum.z + X[i].hiz.z;
196          X[i].konum.u = X[i].konum.u + X[i].hiz.u;
197          X[i].konum.v = X[i].konum.v + X[i].hiz.v;
198          X[i].konum.a = X[i].konum.a + X[i].hiz.a;
199          X[i].konum.b = X[i].konum.b + X[i].hiz.b;
200
201        if( X[i].konum.x < a1 )
202          {
203
204            X[i].konum.x = (b1 - a1)*frand()+ a1;
205            X[i].konum.y = (b2 - a2)*frand()+ a2;
206            X[i].konum.z = (b3 - a3)*frand()+ a3;
207            X[i].konum.u = (b4 - a4)*frand()+ a4;
208            X[i].konum.v = (b5 - a5)*frand()+ a5;
209            X[i].konum.a = (b6 - a6)*frand()+ a6;
```

```
210            X[i].konum.b = (b7 - a7)*frand()+ a7;
211
212            X[i].hiz.x = 0;
213            X[i].hiz.y = 0;
214            X[i].hiz.z = 0;
215            X[i].hiz.u = 0;
216            X[i].hiz.v = 0;
217            X[i].hiz.a = 0;
218            X[i].hiz.b = 0;
219          }
220
221        if( X[i].konum.x > b1 )
222          {
223
224            X[i].konum.x = (b1 - a1)*frand()+ a1;
225            X[i].konum.y = (b2 - a2)*frand()+ a2;
226            X[i].konum.z = (b3 - a3)*frand()+ a3;
227            X[i].konum.u = (b4 - a4)*frand()+ a4;
228            X[i].konum.v = (b5 - a5)*frand()+ a5;
229            X[i].konum.a = (b6 - a6)*frand()+ a6;
230            X[i].konum.b = (b7 - a7)*frand()+ a7;
231
232            X[i].hiz.x = 0;
233            X[i].hiz.y = 0;
234            X[i].hiz.z = 0;
235            X[i].hiz.u = 0;
236            X[i].hiz.v = 0;
237            X[i].hiz.a = 0;
238            X[i].hiz.b = 0;
239
```

```
240         }
241         if( X[i].konum.y < a2 )
242          {
243
244            X[i].konum.x = (b1 - a1)*frand()+ a1;
245             X[i].konum.y = (b2 - a2)*frand()+ a2;
246             X[i].konum.z = (b3 - a3)*frand()+ a3;
247             X[i].konum.u = (b4 - a4)*frand()+ a4;
248             X[i].konum.v = (b5 - a5)*frand()+ a5;
249             X[i].konum.a = (b6 - a6)*frand()+ a6;
250             X[i].konum.b = (b7 - a7)*frand()+ a7;
251
252             X[i].hiz.x = 0;
253             X[i].hiz.y = 0;
254             X[i].hiz.z = 0;
255             X[i].hiz.u = 0;
256             X[i].hiz.v = 0;
257             X[i].hiz.a = 0;
258             X[i].hiz.b = 0;
259
260           }
261         if( X[i].konum.y > b2 )
262          {
263
264             X[i].konum.x = (b1 - a1)*frand()+ a1;
265             X[i].konum.y = (b2 - a2)*frand()+ a2;
266             X[i].konum.z = (b3 - a3)*frand()+ a3;
267             X[i].konum.u = (b4 - a4)*frand()+ a4;
268             X[i].konum.v = (b5 - a5)*frand()+ a5;
269             X[i].konum.a = (b6 - a6)*frand()+ a6;
```

```
270             X[i].konum.b = (b7 - a7)*frand()+ a7;
271
272             X[i].hiz.x = 0;
273             X[i].hiz.y = 0;
274             X[i].hiz.z = 0;
275             X[i].hiz.u = 0;
276             X[i].hiz.v = 0;
277             X[i].hiz.a = 0;
278             X[i].hiz.b = 0;
279           }
280
281
282
283         if( X[i].konum.z < a3 )
284          {
285
286            X[i].konum.x = (b1 - a1)*frand()+ a1;
287             X[i].konum.y = (b2 - a2)*frand()+ a2;
288             X[i].konum.z = (b3 - a3)*frand()+ a3;
289             X[i].konum.u = (b4 - a4)*frand()+ a4;
290             X[i].konum.v = (b5 - a5)*frand()+ a5;
291             X[i].konum.a = (b6 - a6)*frand()+ a6;
292             X[i].konum.b = (b7 - a7)*frand()+ a7;
293
294             X[i].hiz.x = 0;
295             X[i].hiz.y = 0;
296             X[i].hiz.z = 0;
297             X[i].hiz.u = 0;
298             X[i].hiz.v = 0;
299             X[i].hiz.a = 0;
```

```
300                 X[i].hiz.b = 0;
301               }
302             if( X[i].konum.z > b3 )
303               {
304
305                 X[i].konum.x = (b1 - a1)*frand()+ a1;
306                 X[i].konum.y = (b2 - a2)*frand()+ a2;
307                 X[i].konum.z = (b3 - a3)*frand()+ a3;
308                 X[i].konum.u = (b4 - a4)*frand()+ a4;
309                 X[i].konum.v = (b5 - a5)*frand()+ a5;
310                 X[i].konum.a = (b6 - a6)*frand()+ a6;
311                 X[i].konum.b = (b7 - a7)*frand()+ a7;
312
313                 X[i].hiz.x = 0;
314                 X[i].hiz.y = 0;
315                 X[i].hiz.z = 0;
316                 X[i].hiz.u = 0;
317                 X[i].hiz.v = 0;
318                 X[i].hiz.a = 0;
319                 X[i].hiz.b = 0;
320               }
321
322             if( X[i].konum.u < a4 )
323             {
324
325                 X[i].konum.x = (b1 - a1)*frand()+ a1;
326                 X[i].konum.y = (b2 - a2)*frand()+ a2;
327                 X[i].konum.z = (b3 - a3)*frand()+ a3;
328                 X[i].konum.u = (b4 - a4)*frand()+ a4;
329                 X[i].konum.v = (b5 - a5)*frand()+ a5;
```

```
330                 X[i].konum.a = (b6 - a6)*frand()+ a6;
331                 X[i].konum.b = (b7 - a7)*frand()+ a7;
332
333                 X[i].hiz.x = 0;
334                 X[i].hiz.y = 0;
335                 X[i].hiz.z = 0;
336                 X[i].hiz.u = 0;
337                 X[i].hiz.v = 0;
338                 X[i].hiz.a = 0;
339                 X[i].hiz.b = 0;
340             }
341
342             if( X[i].konum.u > b4 )
343               {
344
345                 X[i].konum.x = (b1 - a1)*frand()+ a1;
346                 X[i].konum.y = (b2 - a2)*frand()+ a2;
347                 X[i].konum.z = (b3 - a3)*frand()+ a3;
348                 X[i].konum.u = (b4 - a4)*frand()+ a4;
349                 X[i].konum.v = (b5 - a5)*frand()+ a5;
350                 X[i].konum.a = (b6 - a6)*frand()+ a6;
351                 X[i].konum.b = (b7 - a7)*frand()+ a7;
352
353                 X[i].hiz.x = 0;
354                 X[i].hiz.y = 0;
355                 X[i].hiz.z = 0;
356                 X[i].hiz.u = 0;
357                 X[i].hiz.v = 0;
358                 X[i].hiz.a = 0;
359                 X[i].hiz.b = 0;
```

```
360         }
361
362         if( X[i].konum.u < a5 )
363         {
364
365             X[i].konum.x = (b1 - a1)*frand()+ a1;
366             X[i].konum.y = (b2 - a2)*frand()+ a2;
367             X[i].konum.z = (b3 - a3)*frand()+ a3;
368             X[i].konum.u = (b4 - a4)*frand()+ a4;
369             X[i].konum.v = (b5 - a5)*frand()+ a5;
370             X[i].konum.a = (b6 - a6)*frand()+ a6;
371             X[i].konum.b = (b7 - a7)*frand()+ a7;
372
373             X[i].hiz.x = 0;
374             X[i].hiz.y = 0;
375             X[i].hiz.z = 0;
376             X[i].hiz.u = 0;
377             X[i].hiz.v = 0;
378             X[i].hiz.a = 0;
379             X[i].hiz.b = 0;
380         }
381
382         if( X[i].konum.u > b5 )
383          {
384
385             X[i].konum.x = (b1 - a1)*frand()+ a1;
386             X[i].konum.y = (b2 - a2)*frand()+ a2;
387             X[i].konum.z = (b3 - a3)*frand()+ a3;
388             X[i].konum.u = (b4 - a4)*frand()+ a4;
389             X[i].konum.v = (b5 - a5)*frand()+ a5;
```

```
390             X[i].konum.a = (b6 - a6)*frand()+ a6;
391             X[i].konum.b = (b7 - a7)*frand()+ a7;
392
393             X[i].hiz.x = 0;
394             X[i].hiz.y = 0;
395             X[i].hiz.z = 0;
396             X[i].hiz.u = 0;
397             X[i].hiz.v = 0;
398             X[i].hiz.a = 0;
399             X[i].hiz.b = 0;
400          }
401
402          if( X[i].konum.u < a6 )
403         {
404
405             X[i].konum.x = (b1 - a1)*frand()+ a1;
406             X[i].konum.y = (b2 - a2)*frand()+ a2;
407             X[i].konum.z = (b3 - a3)*frand()+ a3;
408             X[i].konum.u = (b4 - a4)*frand()+ a4;
409             X[i].konum.v = (b5 - a5)*frand()+ a5;
410             X[i].konum.a = (b6 - a6)*frand()+ a6;
411             X[i].konum.b = (b7 - a7)*frand()+ a7;
412
413             X[i].hiz.x = 0;
414             X[i].hiz.y = 0;
415             X[i].hiz.z = 0;
416             X[i].hiz.u = 0;
417             X[i].hiz.v = 0;
418             X[i].hiz.a = 0;
419             X[i].hiz.b = 0;
```

```
420         }
421
422         if( X[i].konum.u > b6 )
423          {
424
425             X[i].konum.x = (b1 - a1)*frand()+ a1;
426             X[i].konum.y = (b2 - a2)*frand()+ a2;
427             X[i].konum.z = (b3 - a3)*frand()+ a3;
428             X[i].konum.u = (b4 - a4)*frand()+ a4;
429             X[i].konum.v = (b5 - a5)*frand()+ a5;
430             X[i].konum.a = (b6 - a6)*frand()+ a6;
431             X[i].konum.b = (b7 - a7)*frand()+ a7;
432
433             X[i].hiz.x = 0;
434             X[i].hiz.y = 0;
435             X[i].hiz.z = 0;
436             X[i].hiz.u = 0;
437             X[i].hiz.v = 0;
438             X[i].hiz.a = 0;
439             X[i].hiz.b = 0;
440          }
441
442          if( X[i].konum.u < a7 )
443         {
444
445             X[i].konum.x = (b1 - a1)*frand()+ a1;
446             X[i].konum.y = (b2 - a2)*frand()+ a2;
447             X[i].konum.z = (b3 - a3)*frand()+ a3;
448             X[i].konum.u = (b4 - a4)*frand()+ a4;
449             X[i].konum.v = (b5 - a5)*frand()+ a5;
```

```
450             X[i].konum.a = (b6 - a6)*frand()+ a6;
451             X[i].konum.b = (b7 - a7)*frand()+ a7;
452
453             X[i].hiz.x = 0;
454             X[i].hiz.y = 0;
455             X[i].hiz.z = 0;
456             X[i].hiz.u = 0;
457             X[i].hiz.v = 0;
458             X[i].hiz.a = 0;
459             X[i].hiz.b = 0;
460         }
461
462         if( X[i].konum.u > b7 )
463          {
464
465             X[i].konum.x = (b1 - a1)*frand()+ a1;
466             X[i].konum.y = (b2 - a2)*frand()+ a2;
467             X[i].konum.z = (b3 - a3)*frand()+ a3;
468             X[i].konum.u = (b4 - a4)*frand()+ a4;
469             X[i].konum.v = (b5 - a5)*frand()+ a5;
470             X[i].konum.a = (b6 - a6)*frand()+ a6;
471             X[i].konum.b = (b7 - a7)*frand()+ a7;
472
473             X[i].hiz.x = 0;
474             X[i].hiz.y = 0;
475             X[i].hiz.z = 0;
476             X[i].hiz.u = 0;
477             X[i].hiz.v = 0;
478             X[i].hiz.a = 0;
479             X[i].hiz.b = 0;
```

```
480          }
481
482          if( f(j+1, X[i].konum.x, X[i].konum.y,X[i].konum.z, X[i].konum.u,
483          X[i].konum.v,X[i].konum.a, X[i].konum.b ) <
484              f( j+1,X[i].pbest.x, X[i].pbest.y,X[i].pbest.z, X[i].pbest.u,
485               X[i].konum.v,X[i].konum.a, X[i].konum.b ))
486             {
487                 X[i].pbest.x = X[i].konum.x;
488                 X[i].pbest.y = X[i].konum.y;
489                 X[i].pbest.z = X[i].konum.z;
490                 X[i].pbest.u = X[i].konum.u;
491                 X[i].pbest.v = X[i].konum.v;
492                 X[i].pbest.a = X[i].konum.a;
493                 X[i].pbest.b = X[i].konum.b;
494                 X[i].hiz.x = 0.;
495                 X[i].hiz.y = 0;
496                 X[i].hiz.z = 0;
497                 X[i].hiz.u = 0;
498                 X[i].hiz.v = 0;
499                 X[i].hiz.a = 0;
500                 X[i].hiz.b = 0;
501             }
502
503       }
504
505         for( i = 0; i < N; i++)
506             X[i].amac = f( j+1, X[i].pbest.x, X[i].pbest.y, X[i].pbest.z, X[i].
507
508        for( i = 0; i < N; i++)
509        {     gb = gbesti(i);
```

```
510             X[i].gbest.x = gb.pbest.x;
511             X[i].gbest.y = gb.pbest.y;
512             X[i].gbest.z = gb.pbest.z;
513             X[i].gbest.u = gb.pbest.u;
514             X[i].gbest.v = gb.pbest.v;
515             X[i].gbest.a = gb.pbest.a;
516             X[i].gbest.b = gb.pbest.b;
517         }
518
519     }
520
521       m =0;
522       for(k = 1; k < N; k++)
523     {
524        if( X[k].amac < min )
525         {
526           min = X[k].amac;
527
528           m = k;
529
530         }
531
532       }
533   amacc = ff( X[m].gbest.x, X[m].gbest.y,X[m].gbest.z, X[m].gbest.u,
534    X[m].gbest.v,X[m].gbest.a, X[m].gbest.b);
535
536    gg1 = g1( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
537     X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );
538
539    gg2 = g2( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
```

```
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg3 = g3( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg4 = g4( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg5 = g5( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg6 = g6( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg7 = g7( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
  X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg8 = g8( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
  X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg9 = g9( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
  X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg10 = g10( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  gg11 = g11( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
   X[m].gbest.v,X[m].gbest.a, X[m].gbest.b );

  if(gg1<=0 && gg2<=0 && gg3<=0 && gg4 <=0 && gg5 <=0&& gg6<=0 && gg7<=0
```

```
  && gg8 <=0 && gg9 <=0 && gg10 <=0 && gg11<=0&& amacc<amacc1)
   {

    fprintf( g, "\n iterasyon = %d, x1 = %f, x2 = %f, x3 = %f, x4 = %f,
     x5 = %f, x6 = %f, x7 = %f, amac = %f ", j, X[m].gbest.x, X[m].gbest.y,
    X[m].gbest.z, X[m].gbest.u, X[m].gbest.v, X[m].gbest.a, X[m].gbest.b,
    amacc);

    fprintf(g,"\n g1 = %f, g2 = %f, g3 = %f, g4 = %f, g5 = %f, g6 = %f,
     g7 = %f, g8 = %f, g9 = %f, g10 = %f, g11 = %f",gg1,gg2,gg3,gg4,gg5,gg6,
     gg7,gg8,gg9,gg10,gg11 );
    fprintf(g, "\n\n-------------------------------------------------\n");

    jj++;

    amacc1 = amacc;  ;



    printf("\n II = %d, \t JJ = %d \t amac = %12.5f ",ii, jj, amacc);
   }

  ii++;
  }while( jj<=100);
 printf("\n Bitiiiiiiiiiiii " );

 getch();
 }
```

```
600  ////////////////////////////
601
602  double f( int kk, double x1, double x2, double x3, double x4, double x5,
603   double x6, double x7)
604  {
605       double f1, h, g111, g22, g33, g44, g55, g66, g77, g88, g99, g1010,
606        g1111 ;
607       double sayac = 0, top = 0, akk;
608
609       f1 = 0.7854*x1*x2*x2*(3.3333*x3*x3 + 14.9334*x3-43.0934)
610            -1.508*x1*(x6*x6+x7*x7)+7.477*(x6*x6*x6+x7*x7*x7)
611            +0.7854*(x4*x6*x6+x5*x7*x7);
612
613
614        g111 = g1 ( x1, x2, x3, x4, x5, x6, x7 );
615         g22 = g2 ( x1, x2, x3, x4, x5, x6, x7 );
616         g33 = g3 ( x1, x2, x3, x4, x5, x6, x7 );
617         g44 = g4 ( x1, x2, x3, x4, x5, x6, x7 );
618         g55 = g5 ( x1, x2, x3, x4, x5, x6, x7 );
619         g66 = g6 ( x1, x2, x3, x4, x5, x6, x7 );
620         g77 = g7 ( x1, x2, x3, x4, x5, x6, x7 );
621         g88 = g8 ( x1, x2, x3, x4, x5, x6, x7 );
622         g99 = g9 ( x1, x2, x3, x4, x5, x6, x7 );
623       g1010 = g10( x1, x2, x3, x4, x5, x6, x7 );
624       g1111 = g11( x1, x2, x3, x4, x5, x6, x7 );
625
626        if( g111 > 0)
627        {
628          sayac++;
629          top = top +g111;
```

```
630          }
631
632        if( g22>0)
633         {
634          sayac++;
635          top = top +g22;
636          }
637        if( g33>0)
638         {
639          sayac++;
640          top = top + g33;
641          }
642        if( g44>0)
643         {
644          sayac++;
645          top = top + g44;
646          }
647         if( g55>0)
648         {
649          sayac++;
650          top = top + g55;
651          }
652         if( g66>0)
653         {
654          sayac++;
655          top = top + g66;
656          }
657         if( g77>0)
658         {
659          sayac++;
```

```
660         top = top + g77;
661        }
662        if( g88>0)
663        {
664         sayac++;
665         top = top + g88;
666        }
667       if( g99>0)
668        {
669         sayac++;
670         top = top + g99;
671        }
672        if( g1010>0)
673        {
674         sayac++;
675         top = top + g1010;
676        }
677        if( g1111>0)
678        {
679         sayac++;
680         top = top + g1111;
681        }
682         h = 10000000*(sayac + top);
683       return( f1 + h );
684 }
685
686 ////////////////////////////////
687
688 PCK gbesti ( int i)
689 {
```

```
690  PCK temp[11];
691  int k, j, jj, kk = 0, m;
692  double min;
693   k = i;
694
695  for( jj =1; jj<=5; jj++)
696  {
697       j = ( N + ( ++ i))% (N);
698       temp[kk++] = X[j];
699       j = ( N + ( -- k))% (N);
700       temp[kk++]= X[j];
701
702   }
703       temp[kk]=X[i];
704    min = temp[0].amac;
705    m = 0;
706    for( jj = 1; jj< kk; jj++ )
707    {
708       if( min > temp[jj].amac )
709        {min = temp[jj].amac; m  = jj; }
710     }
711 return(temp[m]);
712 }
713 ////////////////////////////////
714 double g1 ( double x1, double x2, double x3, double x4, double x5,
715  double x6, double x7 )
716 {
717   return( (27/(x1*x2*x2*x3))- 1);
718 }
```

```
719  //////////////////////
720  double g2 ( double x1, double x2, double x3, double x4, double x5,
721   double x6, double x7 )
722  {
723     return( (397.5/(x1*x2*x2*x3*x3))-1);
724  }
725  //////////////////////
726  double g3 ( double x1, double x2, double x3, double x4, double x5,
727  double x6, double x7 )
728  {
729      return( (1.93*x4*x4*x4/(x2*x3*x6*x6*x6*x6)) -1);
730  }
731  //////////////////////
732  double g4 ( double x1, double x2, double x3, double x4, double x5,
733   double x6, double x7 )
734  {
735     return( (1.93*x5*x5*x5/(x2*x3*x7*x7*x7*x7)) -1);
736  }
737  //////////////////////
738  double g5 ( double x1, double x2, double x3, double x4, double x5,
739   double x6, double x7 )
740  {
741    return ( (1/(110*x6*x6*x6))* sqrt( (745*x4/(x2*x3))*(745*x4/(x2*x3))
742    + 16.9*1000000 ) -1 );
743  }
744  //////////////////////
745  double g6 ( double x1, double x2, double x3, double x4, double x5, double x6,
746  double x7 )
747  {
748     return ( (1/(85*x7*x7*x7))* sqrt ( (745*x5/(x2*x3))*(745*x5/(x2*x3))
```

```
749     +157.5* 1000000) -1 );
750  }
751  //////////////////////
752  double g7 ( double x1, double x2, double x3, double x4, double x5, double x6,
753   double x7 )
754  {
755     return ((x2*x3)/40 -1);
756  }
757  ////////////////////////////
758  double g8 ( double x1, double x2, double x3, double x4, double x5, double x6,
759   double x7 )
760  {
761     return ((5*x2)/x1 -1 );
762  }
763  //////////////////////
764  double g9 ( double x1, double x2, double x3, double x4, double x5, double x6,
765   double x7 )
766  {
767     return (x1/(12*x2) -1 );
768  }
769  //////////////////////
770  double g10 ( double x1, double x2, double x3, double x4, double x5,
771  double x6, double x7 )
772  {
773     return ((1.5*x6 + 1.9)/x4 -1 );
774  }
775  //////////////////////
776  double g11 ( double x1, double x2, double x3, double x4, double x5,
777  double x6, double x7 )
```

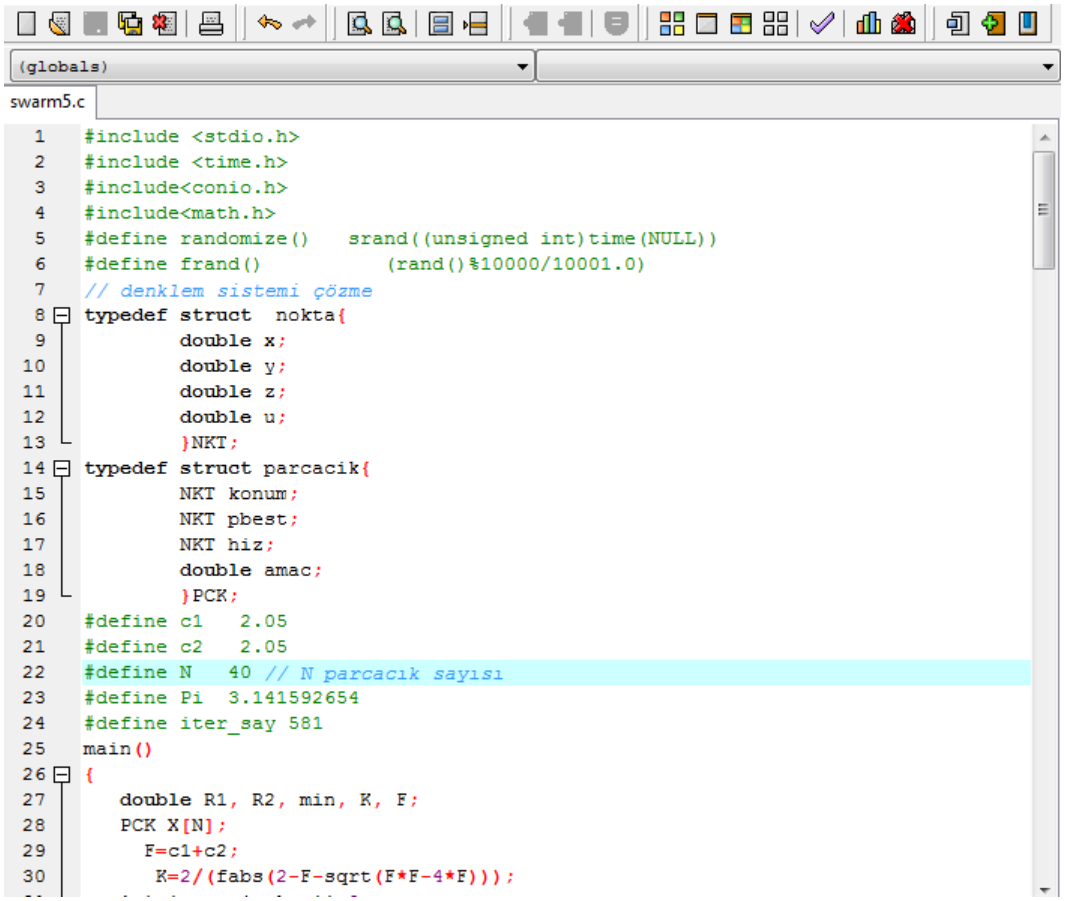
```
778 {
779   return ((1.1*x7 + 1.9)/x5 -1 );
780 }
781
782 ////////////////////
783 double ff( double x1, double x2, double x3, double x4, double x5,
784  double x6, double x7)
785 {
786      double f1;
787
788      f1 = 0.7854*x1*x2*x2*(3.3333*x3*x3 + 14.9334*x3 - 43.0934) -
789      1.508*x1*(x6*x6 + x7*x7)+ 7.4777*(x6*x6*x6 + x7*x7*x7)
790       + 0.7854*(x4*x6*x6 + x5*x7*x7);
791      return( f1);
792 }
793 ////////////////////////////
```

Compiler | Resources | Compile Log | Debug | Find Results

Line: 790 Col: 6 Sel: 0 Lines: 793 Length: 23927 Insert Modified

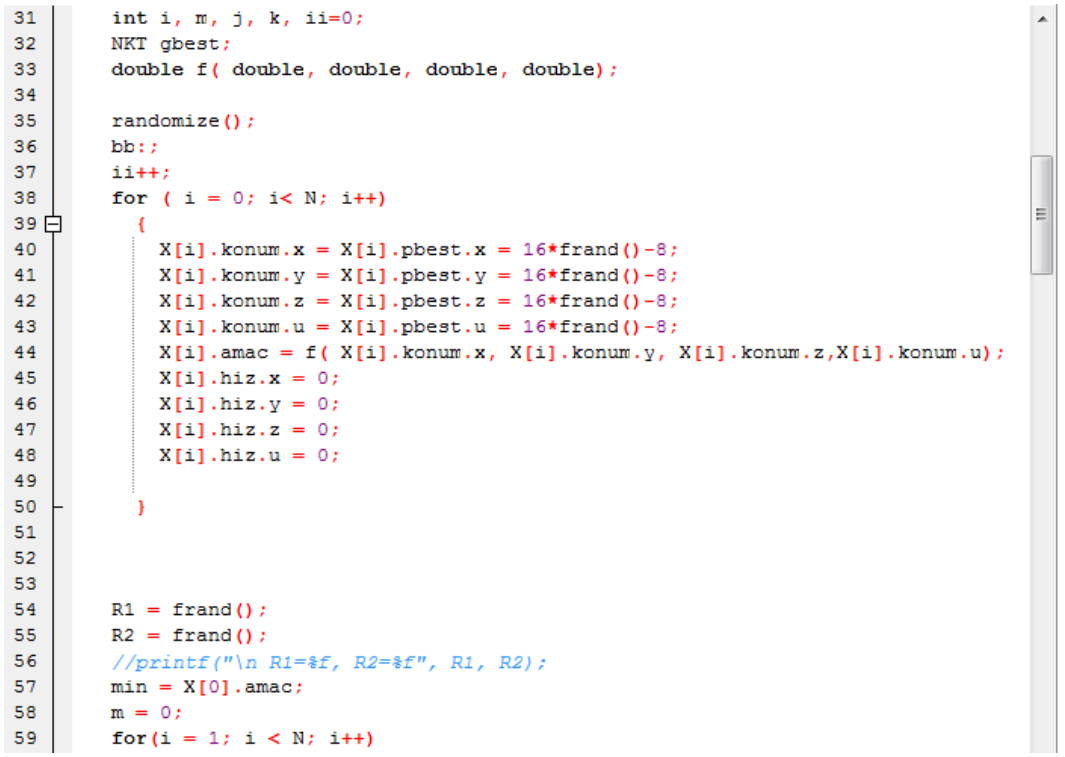
```
II = 3,         JJ = 1          amac =    3485.78936
II = 8,         JJ = 2          amac =    3427.30787
II = 20,        JJ = 3          amac =    3413.92083
II = 34,        JJ = 4          amac =    3338.27792
II = 37,        JJ = 5          amac =    3277.51757
II = 125,       JJ = 6          amac =    3232.62772
II = 133,       JJ = 7          amac =    3207.51614
II = 272,       JJ = 8          amac =    3193.61511
II = 279,       JJ = 9          amac =    3180.36446
II = 363,       JJ = 10         amac =    3161.72695
II = 417,       JJ = 11         amac =    3134.51470
II = 483,       JJ = 12         amac =    3123.93391
II = 941,       JJ = 13         amac =    3122.12174
II = 6538,      JJ = 14         amac =    3113.07256
II = 10291,     JJ = 15         amac =    3108.93557
II = 15382,     JJ = 16         amac =    3082.66645
II = 31300,     JJ = 17         amac =    3017.90251
```

4.3. LİNEER DENKLEM SİSTEMLERİNİN KÖKLERİNİN BULUNMASI



swarm5.c

```
1   #include <stdio.h>
2   #include <time.h>
3   #include<conio.h>
4   #include<math.h>
5   #define randomize()    srand((unsigned int)time(NULL))
6   #define frand()          (rand()%10000/10001.0)
7   // denklem sistemi çözme
8   typedef struct  nokta{
9           double x;
10          double y;
11          double z;
12          double u;
13          }NKT;
14  typedef struct parcacik{
15          NKT konum;
16          NKT pbest;
17          NKT hiz;
18          double amac;
19          }PCK;
20  #define c1    2.05
21  #define c2    2.05
22  #define N    40 // N parcacık sayısı
23  #define Pi   3.141592654
24  #define iter_say 581
25  main()
26  {
27     double R1, R2, min, K, F;
28     PCK X[N];
29       F=c1+c2;
30        K=2/(fabs(2-F-sqrt(F*F-4*F)));
```



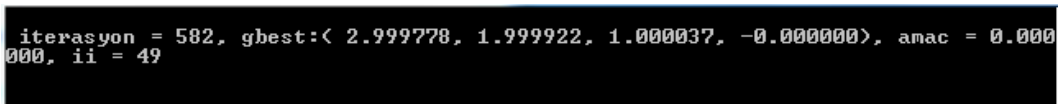
```
31     int i, m, j, k, ii=0;
32     NKT gbest;
33     double f( double, double, double, double);
34
35     randomize();
36     bb:;
37     ii++;
38     for ( i = 0; i< N; i++)
39       {
40         X[i].konum.x = X[i].pbest.x = 16*frand()-8;
41         X[i].konum.y = X[i].pbest.y = 16*frand()-8;
42         X[i].konum.z = X[i].pbest.z = 16*frand()-8;
43         X[i].konum.u = X[i].pbest.u = 16*frand()-8;
44         X[i].amac = f( X[i].konum.x, X[i].konum.y, X[i].konum.z,X[i].konum.u);
45         X[i].hiz.x = 0;
46         X[i].hiz.y = 0;
47         X[i].hiz.z = 0;
48         X[i].hiz.u = 0;
49
50       }
51
52
53
54     R1 = frand();
55     R2 = frand();
56     //printf("\n R1=%f, R2=%f", R1, R2);
57     min = X[0].amac;
58     m = 0;
59     for(i = 1; i < N; i++)
```

```
60      {
61          if( X[i].amac < min )
62          {
63            min = X[i].amac;
64            m = i;
65          }
66       }
67      gbest.x = X[m].konum.x;
68      gbest.y = X[m].konum.y;
69      gbest.z = X[m].konum.z;
70      gbest.u = X[m].konum.u;
71
72
73
74
75       for( j = 1; j<= iter_say; j++)
76      {
77          R1 = frand();
78          R2 = frand();
79      //  printf("\n R1=%f, R2=%f", R1, R2);
80
81
82
83
84         for( i = 0; i < N; i++)
85          {
86            X[i].hiz.x = K*(X[i].hiz.x + c1*R1* ( X[i].pbest.x - X[i].konum.x)
87                        + c2*R2*(gbest.x -  X[i].konum.x ));
88
89            X[i].hiz.y = K*(X[i].hiz.y + c1*R1* ( X[i].pbest.y - X[i].konum.y)
90                        + c2*R2*(gbest.y -  X[i].konum.y ));
91
92            X[i].hiz.z = K*(X[i].hiz.z + c1*R1* ( X[i].pbest.z - X[i].konum.z)
93                        + c2*R2*(gbest.z -  X[i].konum.z ));
94
95            X[i].hiz.u = K*(X[i].hiz.u + c1*R1* ( X[i].pbest.u - X[i].konum.u)
96                        + c2*R2*(gbest.u -  X[i].konum.u ));
97
98
99            X[i].konum.x = X[i].konum.x + X[i].hiz.x;
100           X[i].konum.y = X[i].konum.y + X[i].hiz.y;
101
102           X[i].konum.z = X[i].konum.z + X[i].hiz.z;
103           X[i].konum.u = X[i].konum.u + X[i].hiz.u;
104
105
106           if( f( X[i].konum.x, X[i].konum.y,X[i].konum.z,X[i].konum.u ) <
107               f( X[i].pbest.x, X[i].pbest.y,X[i].pbest.z,X[i].pbest.u ))
108           {
109               X[i].pbest.x = X[i].konum.x;
110               X[i].pbest.y = X[i].konum.y;
111               X[i].pbest.z = X[i].konum.z;
112               X[i].pbest.u = X[i].konum.u;
113           }
114
115
116        }
117
118        for( i = 0; i < N; i++)
119        X[i].amac = f( X[i].pbest.x, X[i].pbest.y, X[i].pbest.z, X[i].pbest.u);
```

```
122       min = X[0].amac;
123       m =0;
124       for(k = 1; k < N; k++)
125     {
126        if( X[k].amac < min )
127         {
128           min = X[k].amac;
129           m = k;
130         }
131      }
132     gbest.x = X[m].pbest.x;
133     gbest.y = X[m].pbest.y;
134     gbest.z = X[m].pbest.z;
135     gbest.u = X[m].pbest.u;
136   //   printf("\n iterasyon = %d, gbest:( %f, %f, %f, %f), amac = %f ", j, gbest.
137   //    gbest.z, gbest.u, min);
138
139
140     }
141    if(min>0.01) goto bb;
142    printf("\n iterasyon = %d, gbest:( %f, %f, %f, %f), amac = %f, ii = %d ", j,
143     gbest.z, gbest.u, min, ii);
144    getch();
145
146 }
147 ////////////////////
148 ////////////////////////////
149 double f( double x1, double x2, double x3, double x4)
150 {
151       double f1, f2, f3, f4, max1, max2, max;
```

```
152       f1 = fabs( 3 - x1 * x3 * x3);
153       f2 = fabs( x3 * sin(Pi / x2) - x3 - x4 );
154       f3 = fabs( -x2*x3*exp(1-x1*x3)+0.2707 );
155       f4 = fabs( 2*x1*x1*x3 - x2*x2*x2*x2*x3 - x2 );
156
157    /*    f1 = fabs( x1*x1 + 2*x2*x2 + cos(x3)-x4*x4);
158       f2 = fabs(3*x1*x1 + x2*x2 + sin(x3)*sin(x3) -x4*x4 );
159       f3 = fabs( -2*x1*x1-x2*x2- cos(x3) +x4*x4 );
160       f4 = fabs(-2*x1*x1 - x2*x2 - cos(x3)*cos(x3) + x4*x4 ); */
161
162
163       if( f1 > f2 )
164         max1 = f1;
165       else
166         max1 = f2;
167
168       if( f3 > f4 )
169         max2 = f3;
170       else
171         max2 = f4;
172
173       if( max1>max2)
174         max = max1;
175       else
176         max = max2;
177
178       return( max);
179 }
```

Compiler | Resources | Compile Log | Debug | Find Results


iterasyon = 582, gbest:(2.999778, 1.999922, 1.000037, -0.000000), amac = 0.000000, ii = 49

4.4. HIMMELBLAU NON-LİNEER OPTİMİZASYON PROBLEMİ

File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

[*] swarm15.c

```c
#include <stdio.h>
#include <time.h>
#include<conio.h>
#include<math.h>
#define randomize()    srand((unsigned int)time(NULL))
#define frand()           (rand()%10000/10001.0)

// himmelblau's nonlinear optimization problem
typedef struct  nokta{
        double x;
        double y;
        double z;
        double u;
        double v;
        }NKT;
typedef struct parcacik{
        NKT konum;
        NKT pbest;
        NKT hiz;
        NKT gbest;
        double amac;
        }PCK;
#define c1   2.885
#define c2   2.885
#define N   36// N parcacık sayısı
#define Pi  3.141592654
#define iter_say  100

PCK X[N];
double g1 ( double, double, double, double, double );
double g2 ( double, double, double, double, double );
double g3 ( double, double, double, double, double );

main()
{
   FILE  *g;
   double R1, R2, min, K, F, vmax , amacc, gg1,gg2,gg3, amacc1 = 300000;
   PCK gb;
   double a1, b1, a2, b2, a3, b3, a4, b4, a5, b5, vmaxx, vmaxy, vmaxz, vmaxu, v
   g = fopen ("dosya15.dat", "w");
   int ii =1;

   F = c1 + c2;
   K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
   int i, m, j, k, jj = 0;
   a1 = 78;
   b1 = 102;

   a2 = 33;
   b2 = 45;

   a3 = 27;
   b3 = 45;

   a4 = 27;
   b4 = 45;

   a5 = 27;
   b5 = 45;
```

```
61    double find_min(int);
62    double find_max( int );
63    PCK gbesti( int ); // ring topolojisini kullanarak r = 3 için i. parcacigin
64
65                            // gbest değerini buluyor
66   randomize();
67    double f( int, double, double, double, double, double );
68     double ff( double x1, double x2, double x3, double x4, double x5 );
69   /*vmaxx = ( b1- a1)/10;
70   vmaxy = ( b2 - a2 )/10;
71   vmaxz = ( b3 - a3 )/10;
72   vmaxu = ( b4 - a4 )/10;*/
73    vmax = ( b1 - a1 );
74    if( vmax > (b2 - a2) ) vmax = b2 - a2;
75    if( vmax > (b3 - a3) ) vmax = b3 - a3;
76    if( vmax > (b4 - a4) ) vmax = b4 - a4;
77    if( vmax > (b5 - a5) ) vmax = b5 - a5;
78    vmax = vmax/2;
79  do
80   {
81
82   for ( i = 0; i< N; i++)
83
84      {
85        X[i].konum.x = X[i].pbest.x = (b1 - a1)*frand()+ a1;
86        X[i].konum.y = X[i].pbest.y = (b2 - a2)*frand()+ a2;
87        X[i].konum.z = X[i].pbest.z = (b3 - a3)*frand()+ a3;
88        X[i].konum.u = X[i].pbest.u = (b4 - a4)*frand()+ a4;
89        X[i].konum.v = X[i].pbest.v = (b5 - a5)*frand()+ a5;
90
```

```
91          X[i].amac = f(2, X[i].konum.x, X[i].konum.y, X[i].konum.z,
92          X[i].konum.u,X[i].konum.v );
93          X[i].hiz.x =
94          X[i].hiz.y =
95          X[i].hiz.z = 0;
96          X[i].hiz.u = 0;
97          X[i].hiz.v = 0;
98       }
99
100
101    R1 = frand();
102    R2 = frand();
103
104
105
106    for(i = 0; i < N; i++)
107       {
108            gb = gbesti(i);
109            X[i].gbest.x = gb.pbest.x;
110            X[i].gbest.y = gb.pbest.y;
111            X[i].gbest.z = gb.pbest.z;
112            X[i].gbest.u = gb.pbest.u;
113            X[i].gbest.v = gb.pbest.v;
114        }
115    for( j = 1; j<= iter_say; j++)
116    {
117
118
119
120      R1 = frand();
```

double f (int kk, double x1, double x2, double x3, double x4,

```
121     R2 = frand();
122
123     for( i = 0; i < N; i++)
124      {
125       X[i].hiz.x = K*(X[i].hiz.x + c1*R1* ( X[i].pbest.x - X[i].konum.x)
126                   + c2*R2*(X[i].gbest.x -  X[i].konum.x ));
127       if( X[i].hiz.x > vmax )  X[i].hiz.x = vmax;
128       if( X[i].hiz.x < -vmax ) X[i].hiz.x = -vmax;
129
130       X[i].hiz.y = K*(X[i].hiz.y + c1*R1* ( X[i].pbest.y - X[i].konum.y)
131                   + c2*R2*(X[i].gbest.y -  X[i].konum.y ));
132
133       if( X[i].hiz.y > vmax )  X[i].hiz.y = vmax;
134       if( X[i].hiz.y < -vmax ) X[i].hiz.y = -vmax;
135
136       X[i].hiz.z = K*(X[i].hiz.z + c1*R1* ( X[i].pbest.z - X[i].konum.z)
137                   + c2*R2*(X[i].gbest.z -  X[i].konum.z ));
138
139       if( X[i].hiz.z > vmax )  X[i].hiz.z =vmax;
140       if( X[i].hiz.z < -vmax ) X[i].hiz.z =-vmax;
141
142       X[i].hiz.u = K*(X[i].hiz.u + c1*R1* ( X[i].pbest.u - X[i].konum.u)
143                   + c2*R2*(X[i].gbest.u -  X[i].konum.u ));
144
145       if( X[i].hiz.u > vmax )  X[i].hiz.u =  vmax;
146       if( X[i].hiz.u < -vmax ) X[i].hiz.u = -vmax;
147
148       X[i].hiz.v = K*(X[i].hiz.v + c1*R1* ( X[i].pbest.v - X[i].konum.v)
149                   + c2*R2*(X[i].gbest.v -  X[i].konum.v ));
150
```

```
151       if( X[i].hiz.v > vmax )  X[i].hiz.v = vmax;
152       if( X[i].hiz.v < -vmax ) X[i].hiz.v = -vmax;
153
154
155       X[i].konum.x = X[i].konum.x + X[i].hiz.x;
156       X[i].konum.y = X[i].konum.y + X[i].hiz.y;
157       X[i].konum.z = X[i].konum.z + X[i].hiz.z;
158       X[i].konum.u = X[i].konum.u + X[i].hiz.u;
159       X[i].konum.v = X[i].konum.v + X[i].hiz.v;
160
161      if( X[i].konum.x < a1 )
162        {
163
164          X[i].konum.x = (b1 - a1)*frand()+ a1;
165          X[i].konum.y = (b2 - a2)*frand()+ a2;
166          X[i].konum.z = (b3 - a3)*frand()+ a3;
167          X[i].konum.u = (b4 - a4)*frand()+ a4;
168          X[i].konum.v = (b5 - a5)*frand()+ a5;
169
170          X[i].hiz.x = 0;
171          X[i].hiz.y = 0;
172          X[i].hiz.z = 0;
173          X[i].hiz.u = 0;
174          X[i].hiz.v = 0;
175        }
176
177      if( X[i].konum.x > b1 )
178       {
179
180         X[i].konum.x = (b1 - a1)*frand()+ a1;
```

```
181             X[i].konum.y = (b2 - a2)*frand()+ a2;
182             X[i].konum.z = (b3 - a3)*frand()+ a3;
183             X[i].konum.u = (b4 - a4)*frand()+ a4;
184             X[i].konum.v = (b5 - a5)*frand()+ a5;
185
186             X[i].hiz.x = 0;
187             X[i].hiz.y = 0;
188             X[i].hiz.z = 0;
189             X[i].hiz.u = 0;
190             X[i].hiz.v = 0;
191          }
192
193          if( X[i].konum.y < a2 )
194           {
195
196            X[i].konum.x = (b1 - a1)*frand()+ a1;
197             X[i].konum.y = (b2 - a2)*frand()+ a2;
198             X[i].konum.z = (b3 - a3)*frand()+ a3;
199             X[i].konum.u = (b4 - a4)*frand()+ a4;
200             X[i].konum.v = (b5 - a5)*frand()+ a5;
201
202             X[i].hiz.x = 0;
203             X[i].hiz.y = 0;
204             X[i].hiz.z = 0;
205             X[i].hiz.u = 0;
206             X[i].hiz.v = 0;
207
208           }
209          if( X[i].konum.y > b2 )
210           {
```

```
211
212             X[i].konum.x = (b1 - a1)*frand()+ a1;
213             X[i].konum.y = (b2 - a2)*frand()+ a2;
214             X[i].konum.z = (b3 - a3)*frand()+ a3;
215             X[i].konum.u = (b4 - a4)*frand()+ a4;
216             X[i].konum.v = (b5 - a5)*frand()+ a5;
217
218             X[i].hiz.x = 0;
219             X[i].hiz.y = 0;
220             X[i].hiz.z = 0;
221             X[i].hiz.u = 0;
222             X[i].hiz.v = 0;
223           }
224
225          if( X[i].konum.z < a3 )
226           {
227
228             X[i].konum.x = (b1 - a1)*frand()+ a1;
229             X[i].konum.y = (b2 - a2)*frand()+ a2;
230             X[i].konum.z = (b3 - a3)*frand()+ a3;
231             X[i].konum.u = (b4 - a4)*frand()+ a4;
232             X[i].konum.v = (b5 - a5)*frand()+ a5;
233
234             X[i].hiz.x = 0;
235             X[i].hiz.y = 0;
236             X[i].hiz.z = 0;
237             X[i].hiz.u = 0;
238             X[i].hiz.v = 0;
239          }
240
```

```
241         if( X[i].konum.z > b3 )
242          {
243
244             X[i].konum.x = (b1 - a1)*frand()+ a1;
245             X[i].konum.y = (b2 - a2)*frand()+ a2;
246             X[i].konum.z = (b3 - a3)*frand()+ a3;
247             X[i].konum.u = (b4 - a4)*frand()+ a4;
248             X[i].konum.v = (b5 - a5)*frand()+ a5;
249
250             X[i].hiz.x = 0;
251             X[i].hiz.y = 0;
252             X[i].hiz.z = 0;
253             X[i].hiz.u = 0;
254             X[i].hiz.v = 0;
255          }
256
257         if( X[i].konum.u < a4 )
258         {
259
260             X[i].konum.x = (b1 - a1)*frand()+ a1;
261             X[i].konum.y = (b2 - a2)*frand()+ a2;
262             X[i].konum.z = (b3 - a3)*frand()+ a3;
263             X[i].konum.u = (b4 - a4)*frand()+ a4;
264             X[i].konum.v = (b5 - a5)*frand()+ a5;
265
266             X[i].hiz.x = 0;
267             X[i].hiz.y = 0;
268             X[i].hiz.z = 0;
269             X[i].hiz.u = 0;
270             X[i].hiz.v = 0;
```

```
271         }
272
273         if( X[i].konum.u > b4 )
274          {
275
276             X[i].konum.x = (b1 - a1)*frand()+ a1;
277             X[i].konum.y = (b2 - a2)*frand()+ a2;
278             X[i].konum.z = (b3 - a3)*frand()+ a3;
279             X[i].konum.u = (b4 - a4)*frand()+ a4;
280             X[i].konum.v = (b5 - a5)*frand()+ a5;
281
282             X[i].hiz.x = 0;
283             X[i].hiz.y = 0;
284             X[i].hiz.z = 0;
285             X[i].hiz.u = 0;
286             X[i].hiz.v = 0;
287
288          }
289
290         if( X[i].konum.v < a5 )
291         {
292
293             X[i].konum.x = (b1 - a1)*frand()+ a1;
294             X[i].konum.y = (b2 - a2)*frand()+ a2;
295             X[i].konum.z = (b3 - a3)*frand()+ a3;
296             X[i].konum.u = (b4 - a4)*frand()+ a4;
297             X[i].konum.v = (b5 - a5)*frand()+ a5;
298
299             X[i].hiz.x = 0;
300             X[i].hiz.y = 0;
```

```
301                 X[i].hiz.z = 0;
302                 X[i].hiz.u = 0;
303                 X[i].hiz.v = 0;
304             }
305
306             if( X[i].konum.v > b5 )
307              {
308
309                 X[i].konum.x = (b1 - a1)*frand()+ a1;
310                 X[i].konum.y = (b2 - a2)*frand()+ a2;
311                 X[i].konum.z = (b3 - a3)*frand()+ a3;
312                 X[i].konum.u = (b4 - a4)*frand()+ a4;
313                 X[i].konum.v = (b5 - a5)*frand()+ a5;
314
315                 X[i].hiz.x = 0;
316                 X[i].hiz.y = 0;
317                 X[i].hiz.z = 0;
318                 X[i].hiz.u = 0;
319                 X[i].hiz.v = 0;
320              }
321
322
323             if( f(j+1, X[i].konum.x, X[i].konum.y,X[i].konum.z, X[i].konum.u, X[i].
324                 f( j+1,X[i].pbest.x, X[i].pbest.y,X[i].pbest.z, X[i].pbest.u, X[i].
325                {
326                     X[i].pbest.x = X[i].konum.x;
327                     X[i].pbest.y = X[i].konum.y;
328                     X[i].pbest.z = X[i].konum.z;
329                     X[i].pbest.u = X[i].konum.u;
330                     X[i].pbest.v = X[i].konum.v;
```

```
331
332                     X[i].hiz.x = 0.;
333                     X[i].hiz.y = 0;
334                     X[i].hiz.z = 0;
335                     X[i].hiz.u = 0;
336                     X[i].hiz.v = 0;
337
338                 }
339
340         }
341
342          for( i = 0; i < N; i++)
343             X[i].amac = f( j+1, X[i].pbest.x, X[i].pbest.y, X[i].pbest.z, X[i].pbe
344
345          for( i = 0; i < N; i++)
346          {     gb = gbesti(i);
347                X[i].gbest.x = gb.pbest.x;
348                X[i].gbest.y = gb.pbest.y;
349                X[i].gbest.z = gb.pbest.z;
350                X[i].gbest.u = gb.pbest.u;
351                X[i].gbest.v = gb.pbest.v;
352          }
353
354      }
355
356     min = X[0].amac;
357        m =0;
358        for(k = 1; k < N; k++)
359     {
360        if( X[k].amac < min )
```

```
361         {
362           min = X[k].amac;
363
364           m = k;
365
366         }
367
368       }
369   amacc = ff( X[m].gbest.x, X[m].gbest.y,X[m].gbest.z, X[m].gbest.u,
370   X[m].gbest.v );
371
372    gg1 = g1( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
373     X[m].gbest.v );
374
375    gg2 = g2( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
376     X[m].gbest.v );
377
378    gg3 = g3( X[m].gbest.x, X[m].gbest.y, X[m].gbest.z, X[m].gbest.u,
379     X[m].gbest.v );
380
381
382    if(0<=gg1 && gg1<=92 && 90<=gg2 && gg2<=110 && 20<=gg3 && gg3<=25
383    && amacc<amacc1)
384     {
385
386      fprintf( g, "\n iterasyon = %d, x1 = %f, x2 = %f, x3 = %f, x4 = %f,
387       x5 = %f, amac = %f ", j, X[m].gbest.x, X[m].gbest.y,
388      X[m].gbest.z, X[m].gbest.u, X[m].gbest.v, amacc);
389      int fprintf (FILE *__stream, const char *__format, ...)
390      fprintf(g,"\n g1 = %f, g2 = %f, g3 = %f ",gg1,gg2,gg3 );
```

```
391      fprintf(g, "\n\n-----------------------------------------\n");
392
393      jj++;
394
395      amacc1 = amacc;  ;
396
397
398
399     printf("\n II = %d, \t JJ = %d \t amac = %12.5f ",ii, jj, amacc);
400     }
401    ii++;
402    }while( jj<=100);
403   printf("\n Bitiiiiiiiiiiii " );
404
405   getch();
406   }
407
408
409   ////////////////////////////
410
411   double f( int kk, double x1, double x2, double x3, double x4, double x5 )
412   {
413        double f1, h, g11, g22, g33 ;
414        double sayac = 0, top = 0, akk;
415
416       // f1 = 5.3578547*x3*x3 + 0.8356891*x1*x5 + 37.2932239*x1 - 40792.141 ;
417
418
419        g11 = g1 ( x1, x2, x3, x4, x5 );
420        g22 = g2 ( x1, x2, x3, x4, x5 );
```

```
421       g33 = g3 ( x1, x2, x3, x4, x5 );
422
423       if( g11 > 0)
424        {
425          sayac++;
426          top = top +g11;
427         }
428       if( g22>0)
429        {
430          sayac++;
431          top = top +g22;
432         }
433       if( g33>0)
434        {
435          sayac++;
436          top = top + g33;
437         }
438
439         akk = pow( kk, 0.4);
440   //      printf("\n akk = %f",akk);
441
442        h = pow(10,akk-1)*pow((sayac + top),2*akk);
443
444        return( f1 + h );
445
446  }
447
448  ////////////////////////////////
449
450  PCK gbesti ( int i)
```

```
451  {
452   PCK temp[11];
453   int k, j, jj, kk = 0, m;
454   double min;
455    k = i;
456
457   for( jj =1; jj<=5; jj++)
458   {
459        j = ( N + ( ++ i))% (N);
460        temp[kk++] = X[j];
461        j = ( N + ( -- k))% (N);
462        temp[kk++]= X[j];
463
464    }
465        temp[kk]=X[i];
466     min = temp[0].amac;
467     m = 0;
468     for( jj = 1; jj< kk; jj++ )
469     {
470        if( min > temp[jj].amac )
471         {min = temp[jj].amac; m  = jj; }
472      }
473  return(temp[m]);
474  }
475  ////////////////////////////////
476  double g1 ( double x1, double x2, double x3, double x4, double x5 )
477  {
478    return( 85.334407 + 0.0056858*x2*x5 + 0.00026*x1*x4 - 0.0022053*x3*x5 );
479  }
480  ////////////////////////
```

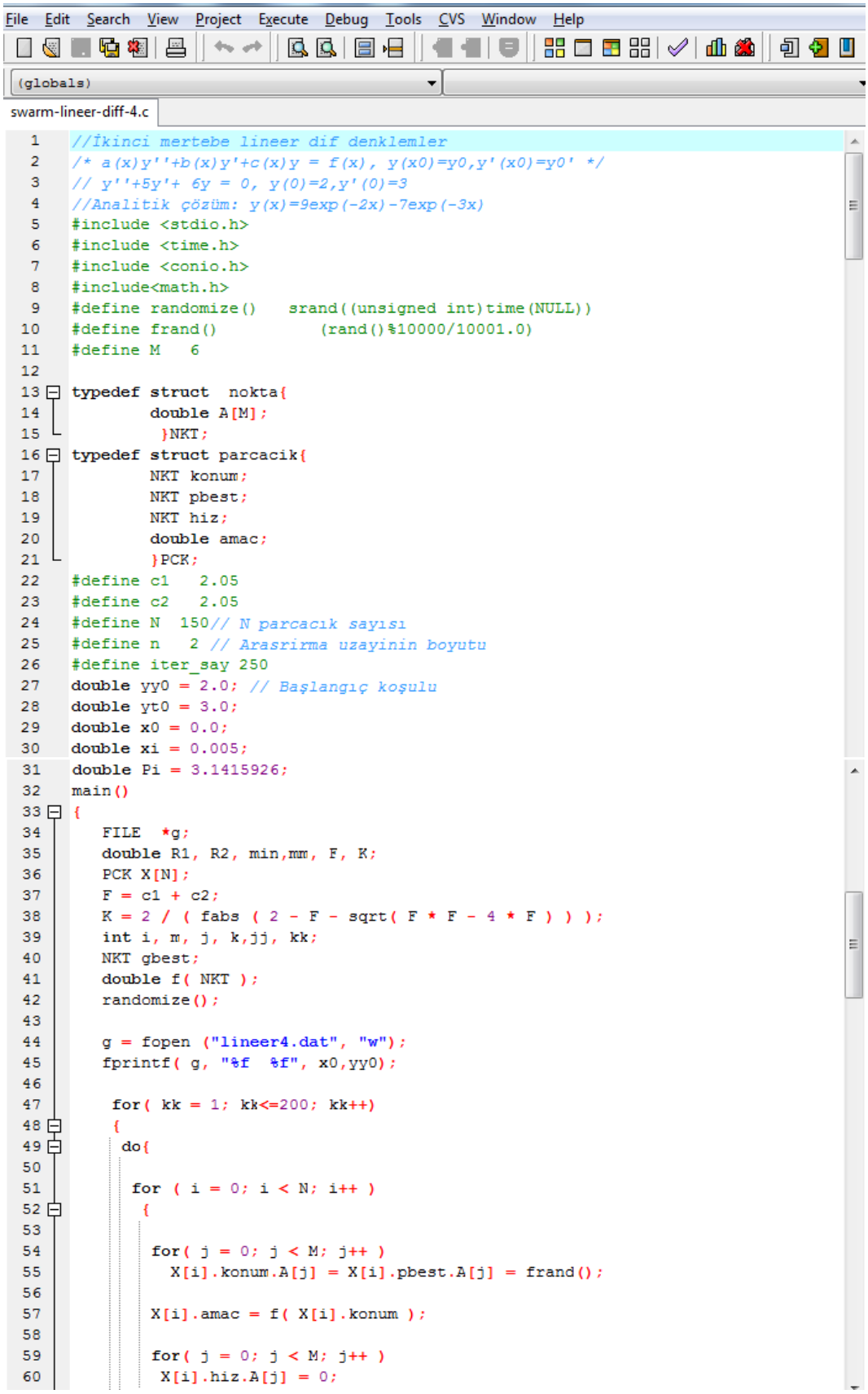
```
481  double g2 ( double x1, double x2, double x3, double x4, double x5 )
482  {
483      return( 80.51249 + 0.0071317*x2*x5 + 0.0029955*x1*x2 + 0.0021813*x3*x3 );
484  }
485  ////////////////////
486  double g3 ( double x1, double x2, double x3, double x4, double x5 )
487  {
488      return( 9.300961 + 0.0047026*x3*x5 + 0.0012547*x1*x3 + 0.0019085*x3*x4 );
489  }
490
491  ////////////////////
492  double ff( double x1, double x2, double x3, double x4, double x5 )
493  {
494       double f1;
495       f1 = 5.3578547*x3*x3 + 0.8356891*x1*x5 + 37.2932239*x1 - 40792.141 ;
496       return( f1);
497  }
498  ////////////////////////////
```

Compiler | Resources | Compile Log | Debug | Find Results

```
II = 12,        JJ = 1          amac = -28894.35437
II = 70,        JJ = 2          amac = -29622.61415
II = 425,       JJ = 3          amac = -29868.63010
II = 1284,      JJ = 4          amac = -30252.43811
```

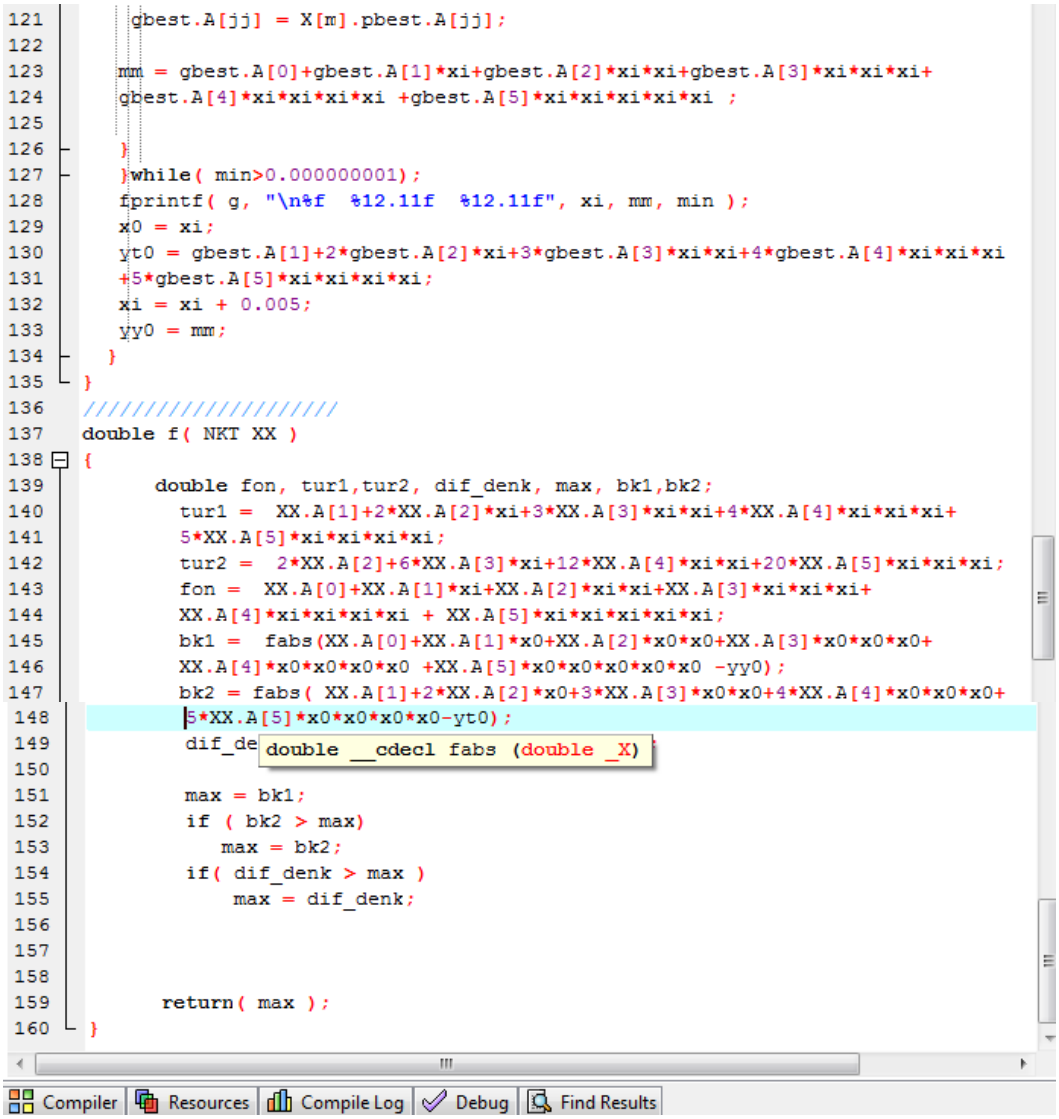
4.5. BAŞLANGIÇ DEĞER PROBLEMLERİ

4.5.1. ÖRNEK 1




```
1  //İkinci mertebe lineer dif denklemler
2  /* a(x)y''+b(x)y'+c(x)y = f(x), y(x0)=y0,y'(x0)=y0' */
3  // y''+5y'+ 6y = 0, y(0)=2,y'(0)=3
4  //Analitik çözüm: y(x)=9exp(-2x)-7exp(-3x)
5  #include <stdio.h>
6  #include <time.h>
7  #include <conio.h>
8  #include<math.h>
9  #define randomize()    srand((unsigned int)time(NULL))
10 #define frand()          (rand()%10000/10001.0)
11 #define M   6
12
13 typedef struct  nokta{
14         double A[M];
15          }NKT;
16 typedef struct parcacik{
17         NKT konum;
18         NKT pbest;
19         NKT hiz;
20         double amac;
21         }PCK;
22 #define c1   2.05
23 #define c2   2.05
24 #define N  150// N parcacık sayısı
25 #define n   2 // Arasrırma uzayinin boyutu
26 #define iter_say 250
27 double yy0 = 2.0; // Başlangıç koşulu
28 double yt0 = 3.0;
29 double x0 = 0.0;
30 double xi = 0.005;
31 double Pi = 3.1415926;
32 main()
33 {
34    FILE  *g;
35    double R1, R2, min,mm, F, K;
36    PCK X[N];
37    F = c1 + c2;
38    K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
39    int i, m, j, k,jj, kk;
40    NKT gbest;
41    double f( NKT );
42    randomize();
43
44    g = fopen ("lineer4.dat", "w");
45    fprintf( g, "%f  %f", x0,yy0);
46
47     for( kk = 1; kk<=200; kk++)
48     {
49      do{
50
51       for ( i = 0; i < N; i++ )
52        {
53
54        for( j = 0; j < M; j++ )
55          X[i].konum.A[j] = X[i].pbest.A[j] = frand();
56
57        X[i].amac = f( X[i].konum );
58
59        for( j = 0; j < M; j++ )
60         X[i].hiz.A[j] = 0;
```

```
61
62
63      R1 = frand();
64      R2 = frand();
65
66      min = X[0].amac;
67      m = 0;
68      for(i = 1; i < N; i++)
69      {
70        if( X[i].amac < min )
71        {
72          min = X[i].amac;
73          m = i;
74        }
75      }
76    for( j = 0 ; j < M; j++ )
77     gbest.A[j] = X[m].konum.A[j];
78
79    for( j = 1; j<= iter_say; j++)
80     {
81      R1 = frand();
82      R2 = frand();
83
84      for( i = 1; i < N; i++)
85       {
86         for( jj = 0; jj < M; jj++)
87          X[i].hiz.A[jj] = K*( X[i].hiz.A[jj] + c1*R1* ( X[i].pbest.A[jj] -
88          X[i].konum.A[jj])+ c2*R2*(gbest.A[jj] -  X[i].konum.A[jj] ));
89
92          for( jj = 0; jj < M; jj++)
93            X[i].konum.A[jj] = X[i].konum.A[jj] + X[i].hiz.A[jj];
94
95
96          if( f( X[i].konum ) <  f( X[i].pbest) )
97             for( jj = 0; jj < M; jj++ )
98                  X[i].pbest.A[jj]  = X[i].konum.A[jj];
99
100
101
102       }
103
104       for( i = 0; i < N; i++)
105         X[i].amac = f( X[i].pbest );
106
107
108      min = X[0].amac;
109      m =0;
110      for(k = 1; k < N; k++)
111       {
112          if( X[k].amac < min )
113          {
114            min = X[k].amac;
115            m = k;
116          }
117       }
118
119
120    for( jj = 0; jj < M; jj++ )
```



```
121     gbest.A[jj] = X[m].pbest.A[jj];
122
123     mm = gbest.A[0]+gbest.A[1]*xi+gbest.A[2]*xi*xi+gbest.A[3]*xi*xi*xi+
124     gbest.A[4]*xi*xi*xi*xi +gbest.A[5]*xi*xi*xi*xi*xi ;
125
126     }
127     }while( min>0.000000001);
128     fprintf( g, "\n%f  %12.11f  %12.11f", xi, mm, min );
129     x0 = xi;
130     yt0 = gbest.A[1]+2*gbest.A[2]*xi+3*gbest.A[3]*xi*xi+4*gbest.A[4]*xi*xi*xi
131     +5*gbest.A[5]*xi*xi*xi*xi;
132     xi = xi + 0.005;
133     yy0 = mm;
134   }
135 }
136 ////////////////////
137 double f( NKT XX )
138 {
139       double fon, tur1,tur2, dif_denk, max, bk1,bk2;
140         tur1 =  XX.A[1]+2*XX.A[2]*xi+3*XX.A[3]*xi*xi+4*XX.A[4]*xi*xi*xi+
141         5*XX.A[5]*xi*xi*xi*xi;
142         tur2 =  2*XX.A[2]+6*XX.A[3]*xi+12*XX.A[4]*xi*xi+20*XX.A[5]*xi*xi*xi;
143         fon =  XX.A[0]+XX.A[1]*xi+XX.A[2]*xi*xi+XX.A[3]*xi*xi*xi+
144         XX.A[4]*xi*xi*xi*xi + XX.A[5]*xi*xi*xi*xi*xi;
145         bk1 =  fabs(XX.A[0]+XX.A[1]*x0+XX.A[2]*x0*x0+XX.A[3]*x0*x0*x0+
146         XX.A[4]*x0*x0*x0*x0 +XX.A[5]*x0*x0*x0*x0*x0 -yy0);
147         bk2 = fabs( XX.A[1]+2*XX.A[2]*x0+3*XX.A[3]*x0*x0+4*XX.A[4]*x0*x0*x0+
148         5*XX.A[5]*x0*x0*x0*x0-yt0);
149         dif_de
150
151         max = bk1;
152         if ( bk2 > max)
153            max = bk2;
154         if( dif_denk > max )
155             max = dif_denk;
156
157
158
159       return( max );
160 }
```

double __cdecl fabs (double _X)

Compiler | Resources | Compile Log | Debug | Find Results

4.5.2. ÖRNEK 2

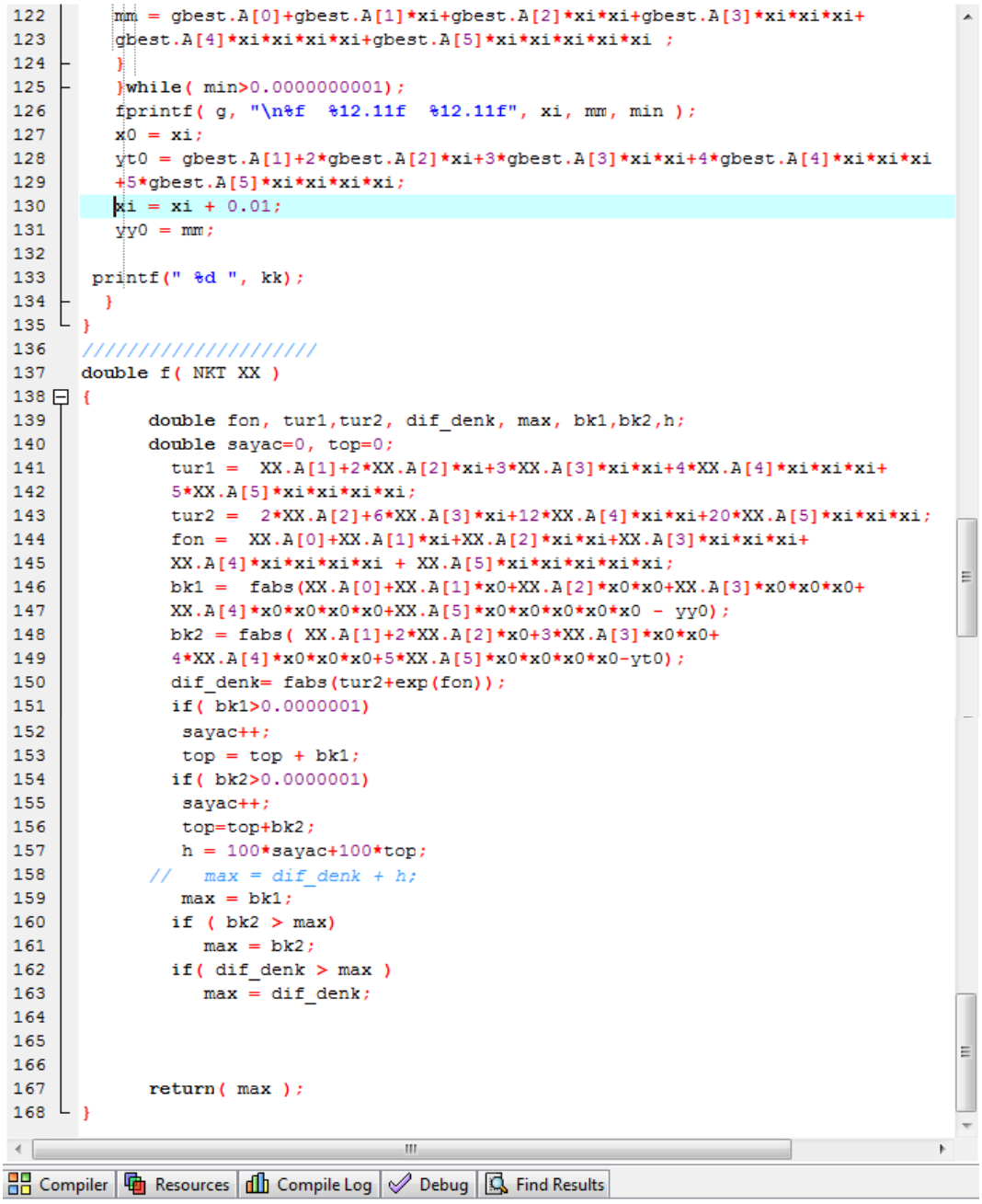
File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

swarm-lineer-diff-7.c

```c
//İkinci mertebe lineer olmayan dif denklemler
// y'' = -Exp(y), y(0)=0,y'(0)=1
// Analitik çözüm y(x) = ln(1+sin(x))
#include <stdio.h>
#include <time.h>
#include <conio.h>
#include<math.h>
#define randomize()    srand((unsigned int)time(NULL))
#define frand()          (rand()%10000/10001.0)
#define M   6

typedef struct  nokta{
        double A[M];
         }NKT;
typedef struct parcacik{
        NKT konum;
        NKT pbest;
        NKT hiz;
        double amac;
        }PCK;
#define c1   2.05
#define c2   2.05
#define N  200// N parcacık sayısı
#define n   2 // Arasrırma uzayinin boyutu
#define iter_say 350
double yy0 = 0.0; // Başlangıç koşulu
double yt0 = 1.0;
double x0 = 0.0;
double xi = 0.01;
double Pi = 3.1415926;
main()
{
   FILE  *g;
   double R1, R2, min,mm, F, K;
   PCK X[N];
   F = c1 + c2;
   K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
   int i, m, j, k,jj, kk;
   NKT gbest;
   double f( NKT );
   randomize();

   g = fopen ("lineer7.dat", "w");
   fprintf( g, "%f  %f", x0,yy0);

    for( kk = 1; kk<=100; kk++)
    {
     do{

      for ( i = 0; i < N; i++ )
       {

       for( j = 0; j < M; j++ )
         X[i].konum.A[j] = X[i].pbest.A[j] = frand();

       X[i].amac = f( X[i].konum );

       for( j = 0; j < M; j++ )
        X[i].hiz.A[j] = 0;
       }
```

```
61
62
63      R1 = frand();
64      R2 = frand();
65
66      min = X[0].amac;
67      m = 0;
68      for(i = 1; i < N; i++)
69      {
70        if( X[i].amac < min )
71        {
72          min = X[i].amac;
73          m = i;
74        }
75      }
76     for( j = 0 ; j < M; j++ )
77      gbest.A[j] = X[m].konum.A[j];
78
79     for( j = 1; j<= iter_say; j++)
80      {
81        R1 = frand();
82        R2 = frand();
83
84       for( i = 1; i < N; i++)
85        {
86          for( jj = 0; jj < M; jj++)
87           X[i].hiz.A[jj] = K*( X[i].hiz.A[jj] + c1*R1* ( X[i].pbest.A[jj] -
88           X[i].konum.A[jj])+ c2*R2*(gbest.A[jj] -  X[i].konum.A[jj] ));
89
95          if( f( X[i].konum ) <  f( X[i].pbest) )
96             for( jj = 0; jj < M; jj++ )
97                  X[i].pbest.A[jj]  = X[i].konum.A[jj];
98
99
100
101       }
102
103       for( i = 0; i < N; i++)
104         X[i].amac = f( X[i].pbest );
105
106
107       min = X[0].amac;
108       m =0;
109       for(k = 1; k < N; k++)
110        {
111          if( X[k].amac < min )
112          {
113            min = X[k].amac;
114            m = k;
115          }
116        }
117
118
119     for( jj = 0; jj < M; jj++ )
120      gbest.A[jj] = X[m].pbest.A[jj];
```



```
122     mm = gbest.A[0]+gbest.A[1]*xi+gbest.A[2]*xi*xi+gbest.A[3]*xi*xi*xi+
123     gbest.A[4]*xi*xi*xi*xi+gbest.A[5]*xi*xi*xi*xi*xi ;
124     }
125     }while( min>0.0000000001);
126     fprintf( g, "\n%f  %12.11f  %12.11f", xi, mm, min );
127     x0 = xi;
128     yt0 = gbest.A[1]+2*gbest.A[2]*xi+3*gbest.A[3]*xi*xi+4*gbest.A[4]*xi*xi*xi
129     +5*gbest.A[5]*xi*xi*xi*xi;
130     xi = xi + 0.01;
131     yy0 = mm;
132
133   printf(" %d ", kk);
134   }
135 }
136 ////////////////////
137 double f( NKT XX )
138 {
139       double fon, tur1,tur2, dif_denk, max, bk1,bk2,h;
140       double sayac=0, top=0;
141         tur1 =  XX.A[1]+2*XX.A[2]*xi+3*XX.A[3]*xi*xi+4*XX.A[4]*xi*xi*xi+
142         5*XX.A[5]*xi*xi*xi*xi;
143         tur2 =  2*XX.A[2]+6*XX.A[3]*xi+12*XX.A[4]*xi*xi+20*XX.A[5]*xi*xi*xi;
144         fon =  XX.A[0]+XX.A[1]*xi+XX.A[2]*xi*xi+XX.A[3]*xi*xi*xi+
145         XX.A[4]*xi*xi*xi*xi + XX.A[5]*xi*xi*xi*xi*xi;
146         bk1 =  fabs(XX.A[0]+XX.A[1]*x0+XX.A[2]*x0*x0+XX.A[3]*x0*x0*x0+
147         XX.A[4]*x0*x0*x0*x0+XX.A[5]*x0*x0*x0*x0*x0 - yy0);
148         bk2 = fabs( XX.A[1]+2*XX.A[2]*x0+3*XX.A[3]*x0*x0+
149         4*XX.A[4]*x0*x0*x0+5*XX.A[5]*x0*x0*x0*x0-yt0);
150         dif_denk= fabs(tur2+exp(fon));
151         if( bk1>0.0000001)
152          sayac++;
153          top = top + bk1;
154         if( bk2>0.0000001)
155          sayac++;
156          top=top+bk2;
157          h = 100*sayac+100*top;
158       //   max = dif_denk + h;
159          max = bk1;
160         if ( bk2 > max)
161            max = bk2;
162         if( dif_denk > max )
163            max = dif_denk;
164
165
166
167       return( max );
168 }
```

Compiler | Resources | Compile Log | Debug | Find Results

4.5.3. ÖRNEK 3

File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

swarm-lineer-diff-8.c

```
1  //İkinci mertebe lineer olmayan dif denklemler
2  // cos(y'')+yy'' = Pi*Pi*x^2/8, y(0)=0,y'(0)=0
3  // Analitik çözüm y(x) = Pi*x^2/4
4  #include <stdio.h>
5  #include <time.h>
6  #include <conio.h>
7  #include<math.h>
8  #define randomize()    srand((unsigned int)time(NULL))
9  #define frand()          (rand()%10000/10001.0)
10 #define M   6
11
12 typedef struct  nokta{
13         double A[M];
14          }NKT;
15 typedef struct parcacik{
16         NKT konum;
17         NKT pbest;
18         NKT hiz;
19         double amac;
20         }PCK;
21 #define c1   2.05
22 #define c2   2.05
23 #define N  200// N parcacık sayısı
24 #define n   2 // Arasrirma uzayinin boyutu
25 #define iter_say 350
26 double yy0 = 0.0; // Başlangıç koşulu
27 double yt0 = 0.0;
28 double x0 = 0.0;
29 double xi = 0.01;
30 double Pi = 3.1415926;
31 main()
32 {
33    FILE  *g;
34    double R1, R2, min,mm, F, K;
35    PCK X[N];
36    F = c1 + c2;
37    K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
38    int i, m, j, k,jj, kk;
39    NKT gbest;
40    double f( NKT );
41    randomize();
42
43    g = fopen ("lineer8.dat", "w");
44    fprintf( g, "%f  %f", x0,yy0);
45
46     for( kk = 1; kk<=100; kk++)
47     {
48      do{
49
50       for ( i = 0; i < N; i++ )
51        {
52
53        for( j = 0; j < M; j++ )
54          X[i].konum.A[j] = X[i].pbest.A[j] = frand();
55
56        X[i].amac = f( X[i].konum );
57
58        for( j = 0; j < M; j++ )
59         X[i].hiz.A[j] = 0;
60       }
```

```
63      R1 = frand();
64      R2 = frand();
65
66      min = X[0].amac;
67      m = 0;
68      for(i = 1; i < N; i++)
69      {
70          if( X[i].amac < min )
71          {
72              min = X[i].amac;
73              m = i;
74          }
75      }
76    for( j = 0 ; j < M; j++ )
77     gbest.A[j] = X[m].konum.A[j];
78
79    for( j = 1; j<= iter_say; j++)
80     {
81        R1 = frand();
82        R2 = frand();
83
84       for( i = 1; i < N; i++)
85        {
86          for( jj = 0; jj < M; jj++)
87           X[i].hiz.A[jj] = K*( X[i].hiz.A[jj] + c1*R1* ( X[i].pbest.A[jj] - X[i
88                        + c2*R2*(gbest.A[jj] -  X[i].konum.A[jj] ));
91          for( jj = 0; jj < M; jj++)
92            X[i].konum.A[jj] = X[i].konum.A[jj] + X[i].hiz.A[jj];
93
94
95          if( f( X[i].konum ) <  f( X[i].pbest) )
96             for( jj = 0; jj < M; jj++ )
97                  X[i].pbest.A[jj]  = X[i].konum.A[jj];
98
99
100
101       }
102
103       for( i = 0; i < N; i++)
104         X[i].amac = f( X[i].pbest );
105
106
107      min = X[0].amac;
108      m =0;
109      for(k = 1; k < N; k++)
110       {
111          if( X[k].amac < min )
112          {
113            min = X[k].amac;
114            m = k;
115          }
116       }
117
118
119    for( jj = 0; jj < M; jj++ )
120     gbest.A[jj] = X[m].pbest.A[jj];
```

```
122     mm = gbest.A[0]+gbest.A[1]*xi+gbest.A[2]*xi*xi+gbest.A[3]*xi*xi*xi+
123     gbest.A[4]*xi*xi*xi*xi +gbest.A[5]*xi*xi*xi*xi*xi ;
124
125     }
126     }while( min>0.0000000001);
127     fprintf( g, "\n%f  %12.11f  %12.11f", xi, mm, min );
128     x0 = xi;
129     yt0 = gbest.A[1]+2*gbest.A[2]*xi+3*gbest.A[3]*xi*xi+
130     4*gbest.A[4]*xi*xi*xi+5*gbest.A[5]*xi*xi*xi*xi;
131     xi = xi + 0.01;
132     yy0 = mm;
133
134   //printf(" %d ", kk);
135    }
136 }
137 ////////////////////
138 double f( NKT XX )
139 {
140       double fon, tur1,tur2, dif_denk, max, bk1,bk2,h;
141       double sayac=0, top=0;
142         tur1 =  XX.A[1]+2*XX.A[2]*xi+3*XX.A[3]*xi*xi+4*XX.A[4]*xi*xi*xi+
143         5*XX.A[5]*xi*xi*xi*xi;
144         tur2 =  2*XX.A[2]+6*XX.A[3]*xi+12*XX.A[4]*xi*xi+20*XX.A[5]*xi*xi*xi;
145         fon =  XX.A[0]+XX.A[1]*xi+XX.A[2]*xi*xi+XX.A[3]*xi*xi*xi+
146         XX.A[4]*xi*xi*xi*xi
147                + XX.A[5]*xi*xi*xi*xi*xi;
148         bk1 =  fabs(XX.A[0]+XX.A[1]*x0+XX.A[2]*x0*x0+XX.A[3]*x0*x0*x0+
149         XX.A[4]*x0*x0*x0*x0+XX.A[5]*x0*x0*x0*x0*x0 - yy0);
150         bk2 = fabs( XX.A[1]+2*XX.A[2]*x0+3*XX.A[3]*x0*x0+
151         4*XX.A[4]*x0*x0*x0+5*XX.A[5]*x0*x0*x0*x0-yt0);
152         dif_denk= fabs(cos(tur2)+tur2*fon-Pi*Pi*xi*xi/8);
153
154          max = bk1;
155         if ( bk2 > max)
156            max = bk2;
157         if( dif_denk > max )
158            max = dif_denk;
159
160
161
162       return( max );
163 }
```

Compiler | Resources | Compile Log | Debug | Find Results

4.5.4. ÖRNEK 4

File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

swarm-lineer-diff-9.c

```
1  //İkinci mertebe lineer olmayan tekil dif denklemler
2  // y''+8y'/x+18y=-4y*ln(y), y(0)=1,y'(0)=0
3  // Analitik çözüm y(x) = Exp(-x^2)
4  #include <stdio.h>
5  #include <time.h>
6  #include <conio.h>
7  #include<math.h>
8  #define randomize()    srand((unsigned int)time(NULL))
9  #define frand()          (rand()%10000/10001.0)
10 #define M   6
11
12 typedef struct  nokta{
13         double A[M];
14          }NKT;
15 typedef struct parcacik{
16         NKT konum;
17         NKT pbest;
18         NKT hiz;
19         double amac;
20         }PCK;
21 #define c1   2.05
22 #define c2   2.05
23 #define N  200// N parcacık sayısı
24 #define n   2 // Arasrırma uzayinin boyutu
25 #define iter_say 350
26 double yy0 = 1.0; // Başlangıç koşulu
27 double yt0 = 0.0;
28 double x0 = 0.0;
29 double xi = 0.01;
30 double Pi = 3.1415926;
31 main()
32 {
33    FILE  *g;
34    double R1, R2, min,mm, F, K;
35    PCK X[N];
36    F = c1 + c2;
37    K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
38    int i, m, j, k,jj, kk;
39    NKT gbest;
40    double f( NKT );
41    randomize();
42
43    g = fopen ("lineer9.dat", "w");
44    fprintf( g, "%f  %f", x0,yy0);
45
46     for( kk = 1; kk<=200; kk++)
47     {
48      do{
49
50      for ( i = 0; i < N; i++ )
51       {
52
53       for( j = 0; j < M; j++ )
54         X[i].konum.A[j] = X[i].pbest.A[j] = frand();
55
56       X[i].amac = f( X[i].konum );
57
58       for( j = 0; j < M; j++ )
59        X[i].hiz.A[j] = 0;
60       }
```

```
63      R1 = frand();
64      R2 = frand();
65
66      min = X[0].amac;
67      m = 0;
68      for(i = 1; i < N; i++)
69      {
70          if( X[i].amac < min )
71          {
72              min = X[i].amac;
73              m = i;
74          }
75      }
76      for( j = 0 ; j < M; j++ )
77      gbest.A[j] = X[m].konum.A[j];
78
79      for( j = 1; j<= iter_say; j++)
80      {
81          R1 = frand();
82          R2 = frand();
83
84          for( i = 1; i < N; i++)
85          {
86              for( jj = 0; jj < M; jj++)
87              X[i].hiz.A[jj] = K*( X[i].hiz.A[jj] + c1*R1* ( X[i].pbest.A[jj] - X[i
88                          + c2*R2*(gbest.A[jj] -  X[i].konum.A[jj] ));
91              for( jj = 0; jj < M; jj++)
92                X[i].konum.A[jj] = X[i].konum.A[jj] + X[i].hiz.A[jj];
93
94
95              if( f( X[i].konum ) <  f( X[i].pbest) )
96                 for( jj = 0; jj < M; jj++ )
97                      X[i].pbest.A[jj]  = X[i].konum.A[jj];
98
99
100
101         }
102
103         for( i = 0; i < N; i++)
104           X[i].amac = f( X[i].pbest );
105
106
107         min = X[0].amac;
108         m =0;
109         for(k = 1; k < N; k++)
110         {
111             if( X[k].amac < min )
112             {
113                 min = X[k].amac;
114                 m = k;
115             }
116         }
117
118
119     for( jj = 0; jj < M; jj++ )
120      gbest.A[jj] = X[m].pbest.A[jj];
```

```
122     mm = gbest.A[0]+gbest.A[1]*xi+gbest.A[2]*xi*xi+gbest.A[3]*xi*xi*xi+
123     gbest.A[4]*xi*xi*xi*xi +gbest.A[5]*xi*xi*xi*xi*xi ;
124      }
125     }while( min>0.0000000001);
126     fprintf( g, "\n%f  %12.11f  %12.11f", xi, mm, min );
127     x0 = xi;
128     yt0 = gbest.A[1]+2*gbest.A[2]*xi+3*gbest.A[3]*xi*xi+
129     4*gbest.A[4]*xi*xi*xi+5*gbest.A[5]*xi*xi*xi*xi;
130     xi = xi + 0.01;
131     yy0 = mm;
132
133   printf(" %d ", kk);
134    }
135 }
136 ////////////////////
137 double f( NKT XX )
138 {
139       double fon, tur1,tur2, dif_denk, max, bk1,bk2,h;
140       double sayac=0, top=0;
141         tur1 =  XX.A[1]+2*XX.A[2]*xi+3*XX.A[3]*xi*xi+4*XX.A[4]*xi*xi*xi+
142         5*XX.A[5]*xi*xi*xi*xi;
143         tur2 =  2*XX.A[2]+6*XX.A[3]*xi+12*XX.A[4]*xi*xi+20*XX.A[5]*xi*xi*xi;
144         fon =  XX.A[0]+XX.A[1]*xi+XX.A[2]*xi*xi+XX.A[3]*xi*xi*xi+
145         XX.A[4]*xi*xi*xi*xi + XX.A[5]*xi*xi*xi*xi*xi;
146         bk1 =  fabs(XX.A[0]+XX.A[1]*x0+XX.A[2]*x0*x0+XX.A[3]*x0*x0*x0+
147         XX.A[4]*x0*x0*x0*x0+XX.A[5]*x0*x0*x0*x0*x0 - yy0);
148         bk2 = fabs( XX.A[1]+2*XX.A[2]*x0+3*XX.A[3]*x0*x0+
149         4*XX.A[4]*x0*x0*x0+5*XX.A[5]*x0*x0*x0*x0-yt0);
150         dif_denk= fabs(tur2+8*tur1/xi+18*fon+4*fon*log(fon));
151
152          max = bk1;
153         if ( bk2 > max)
154            max = bk2;
155         if( dif_denk > max )
156            max = dif_denk;
157
158
159
160       return( max );
161 }
```

Compiler | Resources | Compile Log | Debug | Find Results

4.5.5. ÖRNEK 5

File Edit Search View Project Execute Debug Tools CVS Window Help

(globals)

swarm-lineer-diff-10.c

```
1  //İkinci mertebe lineer olmayan dif denklemler
2  // (y'')^y'+sqrt(y'')=2^(2*x)+sqrt(2
3  // Analitik çözüm y(x) = x^2
4  #include <stdio.h>
5  #include <time.h>
6  #include <conio.h>
7  #include<math.h>
8  #define randomize()    srand((unsigned int)time(NULL))
9  #define frand()           (rand()%10000/10001.0)
10 #define M   6
11
12 typedef struct  nokta{
13         double A[M];
14          }NKT;
15 typedef struct parcacik{
16         NKT konum;
17         NKT pbest;
18         NKT hiz;
19         double amac;
20         }PCK;
21 #define c1   2.05
22 #define c2   2.05
23 #define N  200// N parcacık sayısı
24 #define n   2 // Arasrırma uzayinin boyutu
25 #define iter_say 350
26 double yy0 = 0.0; // Başlangıç koşulu
27 double yt0 = 0.0;
28 double x0 = 0.0;
29 double xi = 0.01;
30 double Pi = 3.1415926;
31 main()
32 {
33    FILE  *g;
34    double R1, R2, min,mm, F, K;
35    PCK X[N];
36    F = c1 + c2;
37    K = 2 / ( fabs ( 2 - F - sqrt( F * F - 4 * F ) ) );
38    int i, m, j, k,jj, kk;
39    NKT gbest;
40    double f( NKT );
41    randomize();
42
43    g = fopen ("lineer10.dat", "w");
44    fprintf( g, "%f  %f", x0,yy0);
45
46     for( kk = 1; kk<=50; kk++)
47     {
48      do{
49
50       for ( i = 0; i < N; i++ )
51        {
52
53        for( j = 0; j < M; j++ )
54          X[i].konum.A[j] = X[i].pbest.A[j] = frand();
55
56        X[i].amac = f( X[i].konum );
57
58        for( j = 0; j < M; j++ )
59         X[i].hiz.A[j] = 0;
60       }
```

```
61
62
63      R1 = frand();
64      R2 = frand();
65
66      min = X[0].amac;
67      m = 0;
68      for(i = 1; i < N; i++)
69      {
70        if( X[i].amac < min )
71        {
72          min = X[i].amac;
73          m = i;
74        }
75      }
76    for( j = 0 ; j < M; j++ )
77     gbest.A[j] = X[m].konum.A[j];
78
79    for( j = 1; j<= iter_say; j++)
80     {
81        R1 = frand();
82        R2 = frand();
83
84       for( i = 1; i < N; i++)
85        {
86          for( jj = 0; jj < M; jj++)
87           X[i].hiz.A[jj] = K*( X[i].hiz.A[jj] + c1*R1* ( X[i].pbest.A[jj] - X[i
88                       + c2*R2*(gbest.A[jj] -  X[i].konum.A[jj] ));
91          for( jj = 0; jj < M; jj++)
92            X[i].konum.A[jj] = X[i].konum.A[jj] + X[i].hiz.A[jj];
93
94
95          if( f( X[i].konum ) <  f( X[i].pbest) )
96             for( jj = 0; jj < M; jj++ )
97                  X[i].pbest.A[jj]  = X[i].konum.A[jj];
98
99
100
101       }
102
103       for( i = 0; i < N; i++)
104         X[i].amac = f( X[i].pbest );
105
106
107      min = X[0].amac;
108      m =0;
109      for(k = 1; k < N; k++)
110       {
111          if( X[k].amac < min )
112          {
113            min = X[k].amac;
114            m = k;
115          }
116       }
117
118
119    for( jj = 0; jj < M; jj++ )
120     gbest.A[jj] = X[m].pbest.A[jj];
```

```
122     mm = gbest.A[0]+gbest.A[1]*xi+gbest.A[2]*xi*xi+gbest.A[3]*xi*xi*xi+
123     gbest.A[4]*xi*xi*xi*xi+gbest.A[5]*xi*xi*xi*xi*xi ;
124
125     }
126     }while( min>1.e-12);
127     fprintf( g, "\n%f  %12.11f  %12.11f", xi, mm, min );
128     x0 = xi;
129     yt0 = gbest.A[1]+2*gbest.A[2]*xi+3*gbest.A[3]*xi*xi+4*
130     gbest.A[4]*xi*xi*xi+5*gbest.A[5]*xi*xi*xi*xi;
131     xi = xi + 0.01;
132     yy0 = mm;
133
134   printf(" %d ", kk);
135    }
136 }
137 ////////////////////
138 double f( NKT XX )
139 {
140       double fon, tur1,tur2, dif_denk, max, bk1,bk2,h;
141       double sayac=0, top=0;
142         tur1 =  XX.A[1]+2*XX.A[2]*xi+3*XX.A[3]*xi*xi+4*XX.A[4]*xi*xi*xi+
143         5*XX.A[5]*xi*xi*xi*xi;
144         tur2 =  2*XX.A[2]+6*XX.A[3]*xi+12*XX.A[4]*xi*xi+20*XX.A[5]*xi*xi*xi;
145         fon =  XX.A[0]+XX.A[1]*xi+XX.A[2]*xi*xi+XX.A[3]*xi*xi*xi+
146         XX.A[4]*xi*xi*xi*xi+ XX.A[5]*xi*xi*xi*xi*xi;
147         bk1 =  fabs(XX.A[0]+XX.A[1]*x0+XX.A[2]*x0*x0+XX.A[3]*x0*x0*x0+
148         XX.A[4]*x0*x0*x0*x0+XX.A[5]*x0*x0*x0*x0*x0 - yy0);
149         bk2 = fabs( XX.A[1]+2*XX.A[2]*x0+3*XX.A[3]*x0*x0+
150         4*XX.A[4]*x0*x0*x0+5*XX.A[5]*x0*x0*x0*x0-yt0);
151         dif_de  double __cdecl fabs (double _X)  2)-pow(2,2*xi)-sqrt(2));
152
153          max = bk1;
154         if ( bk2 > max)
155            max = bk2;
156         if( dif_denk > max )
157            max = dif_denk;
158
159
160
161       return( max );
162 }
```

Compiler | Resources | Compile Log | Debug | Find Results

5. KAYNAKLAR

[1] Parsopoulos, K. E. & Vrahatis, M. N. (2010). *Partial Swarm Optimization and Intelligence Advanced and Application.* New York: Hershey.

[2] Coello, C. A. (2008). *Solving Engineering Optimization Problems with Simple Constrained Particle Swarm Optimizer.* Meksika: Mexico D. F.

[3] Bazaraa, M. S. & Sherali, H. D. & Shetty, C. M. (1979). *Nonlinear Programming Theory and Algorithms.* New York: John Wiley & Sons, Inc

[4] Hu, X. & Eberhart, R. C. & Shi, Y. (2003). *Engineering Optimization with Particle Swarm.* Conferance Publication.