KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

VEKTÖR TABANLI SIFIR YERİ VE EKSTREMUM NOKTA BELİRLEME ALGORİTMALARI

YÜKSEK LİSANS TEZİ

Murat MEMOĞLU

HAZİRAN 2012
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

VEKTÖR TABANLI SIFIR YERİ VE EKSTREMUM NOKTA BELİRLEME ALGORİTMALARI

Murat MEMOĞLU

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"YÜKSEK LİSANS (MATEMATİK)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 25.05.2012
Tezin Savunma Tarihi : 18.06.2012

Tez Danışmanı : Prof. Dr. Erhan COŞKUN

Trabzon 2012

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü

Matematik Ana Bilim Dalında

Murat MEMOĞLU Tarafından Hazırlanan

VEKTÖR TABANLI SIFIR YERİ VE EKSTREMUM NOKTA BELİRLEME ALGORİTMALARI

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 29/05/2012 gün ve 1458/3 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda

YÜKSEK LİSANS TEZİ

olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Erhan COŞKUN ……

Üye : Prof. Dr. Atilla BİLGİN ……

Üye : Yrd. Doç. Dr. Selçuk Han AYDIN ……

Prof. Dr. Sadettin KORKMAZ

Enstitü Müdürü

ÖNSÖZ

Bu çalışmada tek değişkenli fonksiyonların verilen bir aralık içerisindeki bütün reel sıfır yerleri ile iki bilinmeyenli nonlineer denklem sistemlerinin verilen bir bölgedeki bütün reel çözümleri eş zamanlı olarak vektör tabanlı algoritmalar yardımıyla belirlenecektir.

Öncelikle tez konusunun belirlenmesi ve çalışmanın bu hale getirilmesi süresince yardımlarını esirgemeyen Sayın Hocam Prof. Dr. Erhan COŞKUN'a teşekkür eder saygılarımı sunarım.

Ayrıca KTÜ Matematik Bölümü'nün değerli Hocalarına, desteklerini esirgemeyen tüm Araştırma Görevlilerine ve hayatım boyunca hiçbir fedakârlıktan kaçınmayan çok kıymetli aileme teşekkür ederim.

Murat MEMOĞLU

Trabzon 2012

TEZ BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum "Vektör Tabanlı Sıfır Yeri ve Ekstremum Nokta Belirleme Algoritmaları" başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Erhan COŞKUN'un sorumluluğunda tamamladığımı, veri/örnekleri kendim topladığımı deney/analizleri kendim yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecince bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 25/05/2012

Murat MEMOĞLU

İÇİNDEKİLER

Sayfa No

ÖNSÖZ III

TEZ BEYANNAMESİ IV

İÇİNDEKİLER V

ÖZET VII

SUMMARY VIII

ŞEKİLLER DİZİNİ IX

TABLOLAR DİZİNİ X

SEMBOLLER DİZİNİ XI

1. GENEL BİLGİLER 1

1.1 Giriş 1

2. FONKSİYON SIFIR YERLERİ İÇİN VEKTÖR TABANLI YÖNTEMLER 2

2.1. Newton Yöntemi 2

2.2. Sıfır Yerleri İçin Tahminler 5

2.2.1. Sıfır Yeri Tahmin Yönteminin Çözümlemesi 7

2.2.2. Vektörler Üzerinde Aritmetik İşlemler 9

2.2.3. Sıfır Yeri Tahminleri İçin Algoritma 9

2.3. Vektör Tabanlı Newton Yöntemi 10

2.4. Kiriş Yöntemi 14

2.5. Vektör Tabanlı Kiriş Yöntemi 15

3. VEKTÖR TABANLI SAYISAL OPTİMİZASYON YÖNTEMLERİ 19

3.1. Dichotomous Yöntemi 19

3.2. Dik İniş Yöntemi 19

3.3. Yerel Minimumlar İçin Tahminler 20

3.3.1. Yerel Minimum Tahmin Yönteminin Çözümlemesi 21

3.3.2. Yerel Minimumlar İçin Tahmin Algoritması.....22

3.4. Vektör Tabanlı Dichotomous Yöntemi 23

3.5. Tek Değişkenli Fonksiyonların Minimizasyonu İçin Vektör Tabanlı Dik İniş Yöntemi.....25

4. LİNER OLMAYAN DENKLEM SİSTEMLERİNİN SAYISAL ÇÖZÜMLERİ 30

4.1. Denklem Sistemleri İçin Çözüm Tahminleri 30

4.1.1. Tahmin Yönteminin Çözümlemesi 31

4.1.2. Denklem Sistemi Çözüm Tahminleri İçin Algoritma 32

4.2. Denklem Sisteminin Çözümü İçin Vektör Tabanlı Dik İniş Yöntemi 33

5. SONUÇLAR 40

6. ÖNERİLER.....41

7. KAYNAKLAR 42

8. EKLER 43

ÖZGEÇMİŞ

Yüksek Lisans Tezi

ÖZET

VEKTÖR TABANLI SIFIR YERİ VE EKSTREMUM NOKTA BELİRLEME ALGORİTMALARI

Murat MEMOĞLU

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Matematik Anabilim Dalı
Danışman: Prof. Dr. Erhan COŞKUN
2012,42 Sayfa, 12 Ek sayfa

Bu çalışmada tek değişkenli fonksiyonların sıfır yerlerini belirlemek için mevcut Newton ve Kiriş yöntemleri belirli bir aralık üzerindeki bütün reel sıfır yerlerini eş zamanlı olarak bulacak şekilde genelleştirilmiştir. Benzer şekilde iki bilinmeyenli nonlineer denklem sistemleri için Dik İniş yöntemi bir dikdörtgensel bölgedeki bütün reel çözümleri bulacak şekilde genelleştirilmiştir. Ayrıca tek değişkenli fonksiyonların bir açık aralık üzerinde bütün yerel minimumlarını belirlemek için Dichotomous ve Dik İniş yöntemleri genelleştirilmiştir.

Anahtar Kelimeler: Vektör Tabanlı Newton Yöntemi, Vektör Tabanlı Kiriş (Sekant) Yöntemi, Vektör Tabanlı Dichotomous Yöntemi, Vektör Tabanlı Dik İniş Yöntemi.

Master Thesis

SUMMARY

VECTOR BASED ALGORITHMS FOR DETERMINING ZEROS AND EXTREMA POINT

Murat MEMOĞLU

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Mathematics Graduate Program
Supervisor: Prof. Dr. Erhan COŞKUN
2012, 42 Pages, 12 Pages Appendix

The conventional Newton and Secant methods for determining zeros of single variable functions are generalized to determine all the real zeros on a given interval using vector based algorithms. Steepest descent methods is used to determine all the real solutions of a nonlinear system with two unknowns in a given rectangular region. Furthermore, Dichotomous and Steepest descent methods are extended to determine all the extrema of a single variable function on a given interval.

Key Words: Vector-based Newton method, Vector- based secant method, Vector-based Dichotomous method, Vector-based steepest descent method.

ŞEKİLLER DİZİNİ

Sayfa No

Şekil 1. Newton yönteminin grafiksel gösterimi 2

Şekil 2. $f(x)$ fonksiyonu, St sıfır yeri tahminleri ve oluşturulan X ağının grafiği 10

Şekil 3. $f(x) = x - \tan(x)$ fonksiyonu, St sıfır yeri tahminleri ve hesaplanan S sıfır yerlerinin grafiksel gösterimi 14

Şekil 4. Kiriş yöntemi için grafik çizimi 15

Şekil 5. $f(x) = x + e^{-50x^2}\cos(x)$ fonksiyonu, St sıfır yeri tahminleri ve $vkiris$ ile elde edilen S sıfır yeri için grafiksel gösterim.....17

Şekil 6. $f(x) = x^4 - 5x^3 - x^2 + 29x - 26$ fonksiyonu, St sıfır yeri tahminleri ve $vkiris$ ile elde edilen S sıfır yerleri için grafiksel gösterim 18

Şekil 7. $f(x) = \sin(x)$ fonksiyonu, Sta yerel minimum tahmin aralıkları ve oluşturulan X ağı için grafik 23

Şekil 8. $f(x) = -sin(x) - sin(3x)/3$ fonksiyonu için yerel minimumlar.....25

Şekil 9. $f(x) = sinx - xe^{sin(-x)}$ fonksiyonu için yerel minimumlar 27

Şekil 10. $f(x) = sin(x) - xe^{\sin(-x)}$ fonksiyonu için yerel maksimumlar 28

Şekil 11. $f(x) = \tan(x)^2$ fonksiyonu için hesaplanan yerel minimumlar 29

Şekil 12. Oluşturulan XY ağı için grafik 30

Şekil 13. $x^2 + y^2 = 4, x^2 + y^2 - 2x - 3y + 1 = 0$ denklemleri, St çözüm tahminleri ve oluşturulan XY ağı 33

Şekil 14. $x^2 - y - 0.2 = 0, \quad y^2 - x - 0.3 = 0$ denklemleri, St çözüm tahminleri ve S çözüm.....35

Şekil 15. $x^3 - 3x^2 + 4x - y = 0, \ y^2 - x - 2 = 0$ denklemleri, St çözüm tahminleri ve S çözüm.....36

Şekil 16. $y + \sin(x) = 0, \ y + \cos(x) = 0$ denklemleri, St çözüm tahminleri ve S çözüm için grafik.....38

Şekil 17. $2\sin(x) + \sin(y) = 0, \ \sin(x) + 2\sin(y) = 0$ denklemleri, St çözüm tahminleri ve S çözüm için grafik 39

TABLOLAR DİZİNİ

Sayfa No

Tablo 1. $f(x) = x^3 - x^2 - x - 3$ fonksiyonu ve $x_0 = 3$ başlangıç değeri ile Newton iterayonu……..……..3

Tablo 2. $f(x) = 2x - 4 + \sin(2\pi x)$ fonksiyonu ve $x_0 = 2.09$ değeri ile Newton iterasyonu…….....5

SEMBOLLER DİZİNİ

$A \cup B$: A ve B kümelerinin birleşimi

$A \cap B$: A ve B kümelerinin kesişimi

α : Alfa

∞ : Sonsuz

π : Pi sayısı

Δ : Delta

$\in$: Eleman

$\mathbb{R}$: Reel sayılar kümesi

∇F : F'nin gradyanı

$\| \ \|$: Öklid normu

$| \ |$: Mutlak değer

1. GENEL BİLGİLER

1.1. Giriş

Bu çalışmada tek değişkenli fonksiyonların bir aralık içerisindeki bütün sıfır yerleri ile iki bilinmeyenli nonlineer denklem sistemlerinin bir bölgedeki bütün reel sıfır yerlerinin eş zamanlı olarak belirlenmesi hedeflenmiştir. Bu amaçla öncelikle bir nokta komşuluğundaki tek bir sıfır yerini belirleyen Newton ve Kiriş yöntemleri bir aralıktaki tüm reel sıfır yerlerini belirleyecek biçimde genelleştirilmiştir. Söz konusu genelleştirme işlemi vektörel operasyonlar yardımıyla ve eş zamanlı olarak gerçekleştirilmektedir.

Bir sonraki adımda ise açık aralıkta tüm reel sıfır yerlerini belirlemek amacıyla önerilen yöntem iki değişkenli nonlineer denklem sistemlerinin bir diktörtgensel bölgedeki mevcut sıfır yerlerini belirleyecek biçimde genelleştirilmiştir.

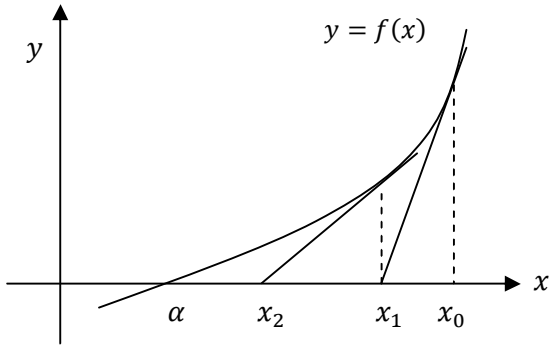
2. FONKSİYON SIFIR YERLERİ İÇİN VEKTÖR TABANLI YÖNTEMLER

2.1. Newton Yöntemi

İteratif yöntemler $f(x) = 0$ denklemini buna denk olan $x = g(x)$ denklemine dönüştürerek, f'nin sıfır yerini bulmak için g'nin sabit noktasını belirlemeye çalışır. Ancak uygun g fonksiyonunun belirlenmesi genelde kolay değildir. Newton yöntemi $g(x) = x - \frac{f(x)}{f'(x)}$ iterasyon fonksiyonunu kullanır ve

$$x_{n+1} = g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)} \qquad n \geq 0 \tag{1}$$

şeklinde iterasyon formülü tanımlar.



Şekil 1. Newton yönteminin grafiksel gösterimi

Geometrik olarak x_1 noktası sıfır yerine yeterince yakın seçilen x_0 noktası için $(x_0, f(x_0))$ noktasından geçen teğet doğrusunun x eksenini kestiği noktadır. İterasyon işlemi x_{n+1} ve x_n arasındaki uzaklığın uygun bir toleranstan küçük kalıncaya kadar devam ettirilir.

Örnek 2.1.1. $f(x) = x^3 - x^2 - x - 3$ fonksiyonuna $x_0 = 3$ başlangıç değeri için Newton yöntemi uygulanarak sıfır yeri bulunursa Tablo 1 elde edilir.

Tablo 1. $f(x) = x^3 - x^2 - x - 3$ fonksiyonu ve $x_0 = 3$ başlangıç değeri ile Newton iterayonu

n	x_n	$f(x_n)$	$\lvert x_{n+1} - x_n \rvert$
0	3	12	
1	2.4	2.664	0.6
2	2.16794425087108	0.32137302	0.23205574
3	2.13127482051864	0.00735140	0.03666943
4	2.13039593336190	$4.165741086E-6$	$8.788871567E-4$
5	2.13039543476744	$1.338484878E-12$	$4.985944617E-7$
6	2.13039543476728	$-1.332267629E-15$	$1.603162047E-13$

f'nin x_n noktası komşuluğunda ikinci mertebeden sürekli türeve sahip olduğunu kabul edelim. Bu durumda x_n noktası komşuluğunda

$$f(x) = f(x_n) + (x - x_n)f'(x_n) + \frac{(x - x_n)^2}{2} f''(c_n) \tag{2}$$

olan bir c_n noktası mevcuttur. $x = \alpha$ olduğunu kabul edersek $f(\alpha) = 0$ olduğundan denklemi α'ya göre çözersek c_n, x ile x_n arasında olacak şekilde

$$\alpha = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{(\alpha - x_n)^2}{2} \cdot \frac{f''(c_n)}{f'(x_n)} \tag{3}$$

denklemi elde edilir. Aslında burada, hata terimi çıkarıldığında elde edilecek yaklaşım (1)' deki ile aynıdır yukarıdaki denklemde $x_n - \frac{f(x_n)}{f'(x_n)}$ terimi yerine x_{n+1} yazılırsa aşağıdaki denklem elde edilir.

$$\alpha - x_{n+1} = -(\alpha - x_n)^2 \frac{f''(c_n)}{2f'(x_n)} \qquad n \geq 0 \tag{4}$$

Tanım 2.1.2. (Yakınsama mertebesi) $\{x_n\}_{n=0}^{n=\infty}$ dizisi α'ya yakınsasın ve $E_n = \alpha - x_n$ olarak tanımlansın. Eğer,

$$\lim_{n\to\infty}\frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^m} = \lim_{n\to\infty}\frac{|E_{n+1}|}{|E_n|^m} = A \tag{5}$$

olacak şekilde $A \neq 0$ ve $m > 0$ sabitleri mevcut ise $\{x_n\}$ dizisi α'ya m'inci mertebeden yakınsar denir. Özel olarak, $m = 1$ ise yakınsama lineer $m = 2$ ise kuadratik olarak adlandırılır.

Genelde Newton yöntemi çok hızlı bir şekilde yakınsar. Ancak, bunun için seçilen başlangıç değerinin köke çok yakın olmasının önemi büyüktür. Yakınsaklık koşulu aşağıdaki teoremde verilmektedir.

Teorem 2.1.3. $f(x)$, $f'(x)$ ve $f''(x)$ α'nın keyfi bir komşuluğunda bütün x değerleri için sürekli ve aynı zamanda $f(\alpha) = 0$, $f'(\alpha) \neq 0$ olsun. Bu takdirde, eğer x_0 başlangıç değeri α'ya yeterince yakın seçilirse, (1) denklemi ile verilen x_n, $n \geq 0$ iterasyonu α'ya yakınsar ve üstelik

$$\lim_{n\to\infty}\frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{f''(\alpha)}{2f'(\alpha)} \tag{6}$$

dir. Bu bağıntı yakınsamanın ikinci mertebeden olduğunu gösterir.

$\{x_n\}$ dizisinin α'ya yakınsaması için x_0 başlangıç noktasını, yeterince küçük bir $I = [\alpha - \varepsilon, \alpha + \varepsilon]$ aralığında,

$$M = \frac{\max_{x\in I}|f''(x)|}{2\min_{x\in I}|f'(x)|} \tag{7}$$

için

$$|\alpha - x_0| < \frac{1}{M} \tag{8}$$

eşitsizliği sağlanacak şekilde seçmemiz yeterli olur (Atkinson, 1989).

Örnek 2.1.4. $f(x) = 2x - 4 + \sin(2\pi x)$ fonksiyonu göz önüne alınırsa $\alpha = 2$ için $f(\alpha) = 0$ dır. $f(x)$, $f'(x)$ ve $f''(x)$ fonksiyonları sürekli, $f'(\alpha) \cong 8.28 \neq 0$ dolayısıyla Teorem 2.1.3.'ün hipotezleri sağlanmaktadır. $\varepsilon = 0.2$ alınırsa $I = [\alpha - \varepsilon, \alpha + \varepsilon] =$

$[1.8, 2.2]$ aralığında $f'(\alpha) \neq 0$, $f'''(1.75) = 0$ dır ve I aralığında $f''(x)$ maksimum değerini bu noktada alır $f'(x)$ de minimum değerini uç noktalarda alır.

$$M = \frac{\max\limits_{x \in I} |f''(x)|}{2 \min\limits_{x \in I} |f'(x)|} = \frac{f''(1.75)}{2f'(1.8)} \cong 10$$

olur dolayısıyla, teoreme göre

$$|\alpha - x_0| < \frac{1}{M} = 0.1 \Rightarrow 1.9 < x_0 < 2.1$$

için Newton iterasyonu $\alpha = 2$ noktasına yakınsar.

Tablo 2. $f(x) = 2x - 4 + \sin(2\pi x)$ fonksiyonu ve $x_0 = 2.09$ değeri ile Newton iterasyonu

n	x_n	$f(x_n)$	$\lvert x_{n+1} - x_n \rvert$
0	2.09	0.71582679	
1	1.99200958	$-6.61650792E-2$	$9.79904246E-2$
2	2.00000510	$4.22117875E-5$	$7.99552068E-3$
3	2	$-9.81573213E-15$	$5.09608151E-6$

Ancak x_0'ı bu aralık dışında aldığımızda bunu garanti edemeyiz. Örneğin $x_0 = 2.25$ alındığında $n = 65$ için $x_{65} = 70.724$ değerini alır.

Giriş kısmında belirtildiği üzere bu çalışmada amacımız öncelikle Newton ve Kiriş yöntemi gibi iteratif yöntemleri, bir aralık üzerindeki bütün sıfır yerleri eş zamanlı olarak bulacak şekilde genelleştirmektir. Söz konusu genelleştirmeleri *vektör tabanlı Newton* ve *vektör tabanlı kiriş* yöntemi olarak adlandıracağız.

Ancak öncelikle vektör tabanlı yöntemler için gerekli vektör değerli başlangıç değerlerini nasıl tahmin ettiğimizi inceleyelim.

2.2. Sıfır Yerleri İçin Tahminler

Kullanıcı tarafından verilen bir (a, b) aralığında tek değişkenli f fonksiyonun bütün reel sıfır yerlerini tahmin etmek istiyoruz.

Kullanıcı aralık için a ve b sınır değerlerini, eğer başlangıç hassasiyetini belirlemek isterse verilen aralıkta oluşturulacak ağdaki en büyük artım miktarı olan Δx'i ve f

fonksiyonunu verir. Yöntem bu bölgedeki bütün reel sıfır yerleri için birer tahmin bulmayı amaçlar.

Öncelikle eğer Δx verilmemiş ise aralık uygun bir N için N adet parçaya bölünerek Δx elde edilir. Yani,

$$\Delta x = \frac{b-a}{N} \tag{9}$$

olarak alınır. Sonrasında fonksiyonun bu aralıktaki sıfır yerlerini tahmin etmek için bir ağ oluşturulur. Bu ağ, fonksiyonun sıfıra yaklaştığı yerlerde sıklaşacak şekilde seçilmelidir. Bunun için adaptif ağ aşağıdaki şekilde oluşturulabilir.

$$x_{i+1} = x_i + \Delta x \frac{(1 + c|f(x_i)|)}{(M + c|f(x_i)|)} \ , \ \ x_0 = a, M > 1, c > 1 \tag{10}$$

Buradan açıkça görülmektedir ki adım uzunluğu en küçük değeri olan $\Delta x/M$'yi $f(x) = 0$ olduğunda alır ve fonksiyon mutlak değerce arttığında bu adım uzunluğu Δx'e yaklaşır.

Daha sonra f'nin X ağındaki değerleri hesaplanarak $Y = |f(X)|$ vektörü elde edilir. Ayrıca, $|f(X)|$ fonksiyonu için Y_p ileri fark vektörü

$$Y_p = Y(2{:}n) \ - Y(1{:}n-1), \qquad n = boyut(X) \tag{11}$$

ve $Y_{p_i} = sgn(Y_p)$ işaret vektörü hesaplanır. Böylece $|f(X)|$ fonksiyonun artan veya azalan olduğu aralıklar belirlenmiş olur. $f(x)$ fonksiyonun sıfır yerleri bu aralıkların değişim noktasında ortaya çıkar.

Notasyon 2.2.1. Herhangi bir X vektörü için $fark$ operatörü aşağıdaki şekilde tanımlanır.

$fark(X) = X(2{:}n) - X(1{:}n-1),\ n = boyut(X)$

Y_{p_i} işaret vektöründe belirlenen aralıkların değişim noktalarını belirlemek için,

$$test = fark\left(Y_{p_i}\right) \tag{12}$$

vektörü hesaplanır. Sonuç olarak $test$ vektöründe sıfırdan farklı indisler belirlenerek bu indislerin ağdaki karşılıkları çözüm tahmini olarak alınır.

2.2.1. Sıfır Yeri Tahmin Yönteminin Çözümlemesi

1. f, a, b ve Δx verileri alınır.
2. Eğer Δx verilmemiş ise $\Delta x = (b-a)/N$ alınır.
3. $x_0 = a$, $x_{son} = b$ olmak üzere $X = [x_i]$ ağı oluşturulur.

$$x_{i+1} = x_i + \Delta x \frac{(1 + c|f(x_i)|)}{(M + c|f(x_i)|)}$$

4. $Y = |f(X)|$ hesaplanır.
5. $Y_p = Y(2{:}n) - Y(1{:}n-1)$, $n = boyut(X)$ hesaplanır.
6. Y_p'nin işaretlerinden oluşan Y_{p_i} vektörü oluşturulur.
7. Y_{p_i}'de işaret değişen noktaların indisleri bulunur.
8. Bulunan indislerin X ağındaki karşılıkları çözüm tahmini olarak alınır.
9. Bulunan çözüm tahminleri geri gönderilir.

Örnek 2.2.2. $f(x) = (\mathrm{x}-1)(\mathrm{x}+3)$, $a = -5$ ve $b = 5$ olsun. Bu durumda, $N = 10$, $M = 10$, $c = 2$ için

$$\Delta x = \frac{b-a}{N} = 1$$

olur ve ardından X ağını

$$x_{i+1} = x_i + \Delta x \frac{(1 + c|f(x_i)|)}{(M + c|f(x_i)|)} \ , \quad x_0 = -5, \ \ i = 0,1, \dots$$

formülü için oluşturursak

$$X = [-5.0000 \quad -4.2647 \quad -3.6507 \quad -3.2114 \quad -2.9754 \quad -2.8581$$
$$-2.6692 \quad -2.3934 \quad -2.0310 \quad -1.5979 \quad -1.1186 \quad -0.6194$$
$$-0.1276 \ 0.3263 \quad 0.7048 \quad 0.9663 \quad 1.0897 \quad 1.2513 \ 1.5097$$
$$1.8932 \quad 2.4130 \quad 3.0572 \quad 3.7994 \quad 4.6122 \quad 5.0000]$$

ve

$$Y_{p_i} = sgn(Y_p) =$$
$$= [-1\ , -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, 1, 1, 1\ , 1\ , 1, 1, 1, 1, 1]$$

olur. Yukarıdaki gibi birden çok aynı değere sahip vektörlerde notasyonel kolaylık için

$$Y_{p_i} = [4(-1)6(1)5(-1)9(1)]$$

şeklinde gösterim kullanıyoruz.

$$test = fark(Y_{p_i}) = [3(0)1(2)5(0)1(-2)4(0)1(2)8(0)]$$

olur. $test$'de ikinci, yedinci ve onuncu değerler sıfırdan farklıdır yani buralarda sıfır yeri arıyoruz.

$$X(4) = [-3.2114]$$

$$X(10) = [-1.5979]$$

$$X(15) = [\ 0.7048]$$

sonuç olarak çözüm tahmini

$$St = [-3.2114 \quad -1.5979 \quad 0.7048]$$

dır. Aslında çözüm tahmininde sıfır yerlerine ilaveten ekstremum noktaları için de tahminler vardır.

Örnek 2.2.3. $f(x) = x^2$, $a = -2$ ve $b = 2$ olsun. Bu durumda, $N = 4$, $M = 5$, $c = 2$ için

$$\Delta x = \frac{b-a}{N} = 1$$

olur ve ardından X ağını

$$x_{i+1} = x_i + \Delta x \frac{(1 + c|f(x)|)}{(M + c|f(x)|)}\ , \quad x_0 = -2, \qquad i = 0,1, \ldots$$

formülü için oluşturursak

$$X = [-2 \quad -1.3077 \quad -0.7827 \quad -0.4253 \quad -0.1713 \quad 0.0380$$
$$0.2384 \quad 0.4562 \quad 0.7177 \quad 1.0544 \quad 1.5006 \quad 2]$$

$$Y_{p_i} = sgn(Y_p) = [5(-1)6(1)],$$

ve

$$test = fark\left(Y_{p_i}\right) = [4(0)1(2)5(0)]$$

olur. Sonuç olarak sıfır yeri tahminin indisi

$$Sti = indis(test > 0) = [5\ ]$$

olur ve sıfır yeri tahminini

$$St = X(Sti) = [-0.1713]$$

şeklinde hesaplarız.

2.2.2. Vektörler Üzerinde Aritmetik İşlemler

Vektörler üzerinde toplama, çıkarma çarpma ve üs alma işlemlerini ve bunların noktasal versiyonlarını aşağıdaki gibi tanımlıyoruz.

$X = [x_1,\ x_2,\ x_3, \dots\ x_n]$, $Y = [y_1,\ y_2,\ y_3, \dots\ y_n]$ iki vektör ve k bir skaler olsun bu durumda

$X + Y = [x_1 + y_1,\ x_2 + y_2,\ x_3 + y_3, \dots, x_n + y_n]$

$X - Y = [x_1 - y_1,\ x_2 - y_2,\ x_3 - y_3, \dots, x_n - y_n]$

$X.* Y = [x_1 y_1,\ x_2 y_2,\ x_3\ y_3, \dots, x_n y_n]$

$X./Y = [x_1/y_1,\ x_2/y_2,\ x_3/\ y_3, \dots, x_n/y_n]$

$X.\hat{}Y = [(x_1)^{y_1}, (\ x_2)^{y_2}, (\ x_3)^{\ y_3}, \dots, (x_n)^{y_n}]$

$X.\hat{}k = [(x_1)^k, (\ x_2)^k, (\ x_3)^k, \dots, (x_n)^k]$

dir.

2.2.3. Sıfır Yeri Tahminleri İçin Algoritma

Söz dizimi : $St = fsıfırtahmin(f, a, b, \Delta x)$

Girdi: f : Sıfır yerleri belirlenecek fonksiyon

a : Aralığın sol uç noktası

b : Aralığın sağ uç noktası

Δx : En büyük adım uzunluğu

1. $i = 1;\ N = 20;\ M = 100;\ c = 5;\ X = a$; başlangıç indis değerleri ve parametreler
2. Eğer Δx verilmemiş ise $\Delta x = (b - a)/N$;
3. $X(son) < b$ olduğu sürece

   3.1. $x1 = X(i) + \Delta x \frac{(1+c|f(X(i))|)}{(M+c|f(X(i))|)}$; adaptif ağ formülünü uygula

   3.2. $X = [X, x1]$;

   3.3. $i = i + 1$;
4. Eğer $X(son) < b$ ise $X = [X, b]$ değilse $X(son) = b$ al;
5. $Y = |f(X)|$;
6. $Yp = fark(Y)$;
7. $Ypi = işaret(Yp)$; Yp'nin işaret vektörü
8. $test = fark(Ypi)$; işaret değişimi

9. $Sti = indis(test \neq 0)$; işaret değişim noktalarının indisleri

10. $St = X(Sti)$; işaret değişim noktalarını sıfır yeri tahmini olarak al

Çıktı: St sıfır yeri tahminleri

Örnek 2.2.4. $f(x) = (x - 0.5)(x - 1.7)(x - 2)$ fonksiyonunun $(0,3)$ aralığındaki sıfır yerleri için tahminler belirleyelim. Ek de verilen $fsifirtahmin$ programını çalıştırırsak aşağıdaki sonuçları elde ederiz.

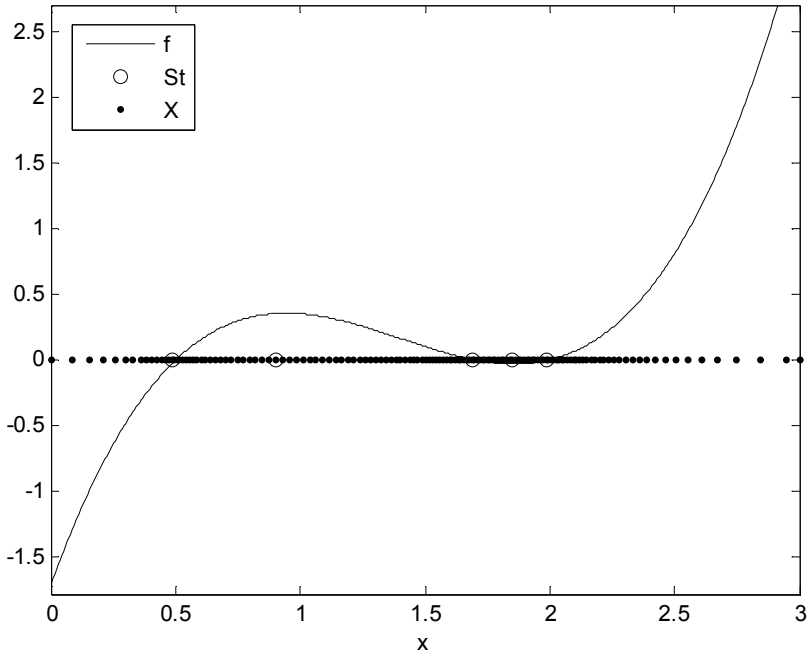
$a = 0$

$b = 3$

$\Delta x = 1$

$>> St = fsifirtahmin('(x - 0.5) * (x - 1.7) * (x - 2)', 0,3,1)$

$St = [0.4855 \quad 0.9020 \quad 1.6881 \quad 1.8506 \quad 1.9847]$



Şekil 2. $f(x)$ fonksiyonu, St sıfır yeri tahminleri ve oluşturulan X ağının grafiği

2.3. Vektör Tabanlı Newton Yöntemi

Newton yöntemi ile $f(x) = 0$ denkleminin birden fazla kökünün sayısal çözümünü hesaplamak için, yöntemde her bir başlangıç değeri ile yeni döngü oluşturmak yerine, daha önce belirlenen vektör değerli başlangıç değerlerini kullanarak, tek bir döngüde bütün başlangıç değerleri için tek işlem yapmak programlama açısından işlem tasarrufu

sağlayacaktır. Buradan hareketle MATLAB ve OCTAVE için kullanacağımız vektör tabanlı Newton algoritmasını aşağıdaki gibi veriyoruz.

Söz dizimi : $X = vnewton(f, X0, a, b, df, imax, eps)$

Girdi: f : Sıfır yerleri belirlenecek fonksiyon

$X0$: Sıfır yerleri için $fsifirtahmin$ ile belirlenen başlangıç vektörü

a : Aralığın sol uç noktası

b : Aralığın sağ uç noktası

df : f'nin türevi

$imax$: Maksimum iterasyon sayısı

eps : Sonuçlandırma kriteri için tolerans

1. $i = 0$; $X = []$; Başlangıç indis değerleri ve parametreler
2. $i < imax$ ve $X0 \neq []$ olduğu sürece 3-8 adımları gerçekleştiriniz.
3.
   3.1. $X0$ vektörünün $df(X0) = 0$ olan indislerini belirle ve i_df0'a ata;
   3.2. $X0$ vektörünün $df(X0) \neq 0$ olan indislerini bul ve if_df0'a ata;
   3.3. $X0$ vektörünün $f(X0) = 0$ olan indislerini bul ve $if0$'a ata;
   3.4. $X0$ vektörünün $df(X0)$'ı sonlu yapan indislerini bul ve is_df0'a ata;
4. Türev ve fonksiyonun aynı anda sıfır olduğu indislerin $X0$ değerlerini X çözüm vektörüne ata
5. $X0 = X0(if\_df0 \cap is\_df0)$ olarak al;
6. $X1 = X0 - f(X0)./df(X0)$ vektörel Newton yaklaşımını hesapla;
7. Yakınsayan bileşenleri çözüme ata, yakınsamayanlarla devam et;
   7.1. $fark = |X1 - X0|$ vektörünü hesapla;
   7.2. $fark > eps$ olan indisleri belirle ve ih vektörüne ata;
   7.3. $fark \leq eps$ olan indisleri belirle ve ie vektörüne ata;
   7.4. $X1(ie)$ bileşenlerini X çözüm vektörüne ata;
   7.5. $X0 = X0(ih)$
8. $i = i + 1$;
9. X'de tekrarlanan değerleri ve (a, b) aralığının dışına çıkanları ayıkla;

Çıktı: X sıfır yerleri

Örnek 2.3.1. Chebyshev polinomlarının sıfır yerlerini bulalım. Chebyshev polinomun genel hali $f(x) = T_n(x) = \cos(narccos(x))$ dir. $n = 5$ için $(-1,1)$ aralığında bulunan kökleri bulalım.

$a = -1$

$b = 1$

$\Delta x = 1$

için

$>> St = fsifirtahmin('cos(5 * acos(x))', a, b, \Delta x)$

$$St = \begin{bmatrix} -1.00000000000000 \\ -0.85909781208334 \\ -0.60393314166611 \\ -0.37681120629743 \\ -0.01081349951811 \\ 0.24649137883048 \\ 0.58042152204463 \\ 0.73838966886344 \\ 0.94032196556926 \end{bmatrix}$$

$df = 5 * sin(5 * acos(x))/(1 - x\text{^}2)\text{^}(1/2)$

$imax = 20$

$eps = 10\text{^} - 7$

$>> S = vnewton('cos(5 * acos(x))', St, a, b, df, imax, eps)$

$$S = \begin{bmatrix} -0.95105651629515 \\ -0.58778525229247 \\ 0.00000000000000 \\ 0.58778525229247 \\ 0.95105651629515 \end{bmatrix}, \quad f(S) = 10^{-14} * \begin{bmatrix} -0.12252657797839 \\ -0.04286263797016 \\ 0.03061616997868 \\ -0.01836970198721 \\ -0.08269460797428 \end{bmatrix}$$

elde ederiz.

Örnek 2.3.2. Legendre polinomlarının sayısal integralde sıkça kullanılan köklerini verilen yöntemle bulmak mümkündür. Örneğin 5-inci dereceden Legendre polinomunun köklerini bulalım.

$a = -1$

$b = 1$

$\Delta x = 0.5$

için

$>> St = fsifirtahmin('1/8 * (63 * x\text{^}5 - 70 * x\text{^}3 + 15 * x)', a, b, \Delta x)$

$$St = \begin{bmatrix} -0.91041747967334 \\ -0.77967966103328 \\ -0.54305759878164 \\ -0.30235822224969 \\ -0.00314384442276 \\ 0.27303384989991 \\ 0.53126496690159 \\ 0.74495644800819 \\ 0.90325370999099 \end{bmatrix}$$

$df = 315/8 * x\text{^}4 - 105/4 * x\text{^}2 + 15/8$

$imax = 20$

$eps = 10\text{^} - 7$

$>> S = vnewton('1/8 * (63 * x\text{^}5 - 70 * x\text{^}3 + 15 * x)', St, a, b, df, imax, eps)$

$$S = \begin{bmatrix} -0.90617984593867 \\ -0.53846931010568 \\ 0 \\ 0.53846931010568 \\ 0.90617984593866 \end{bmatrix}, \quad f(S) = 10^{-13} * \begin{bmatrix} -0.53734794391858 \\ -0.00222044604925 \\ 0 \\ 0.00222044604925 \\ 0.04218847493576 \end{bmatrix}$$

elde ederiz.

Örnek 2.3.3. $f(x) = x - \tan(x)$ fonksiyonunun $(-1,5)$ aralığındaki sıfır yerlerini bulalım bunun için öncelikle $fmintahmin$ programını sonra elde edilen St tahminleri için $vnewton$'u çalıştıralım.

$a = -1$

$b = 5$

$\Delta x = 0.5$

$>> St = fsifirtahmin('x - tan(x)', a, b, \Delta x)$

$$St = \begin{bmatrix} -0.00429139471020 \\ 1.45551966454000 \\ 4.48576143482815 \\ 4.63964894598480 \end{bmatrix}$$

olur.

$df = -tan(x)\text{^}2$

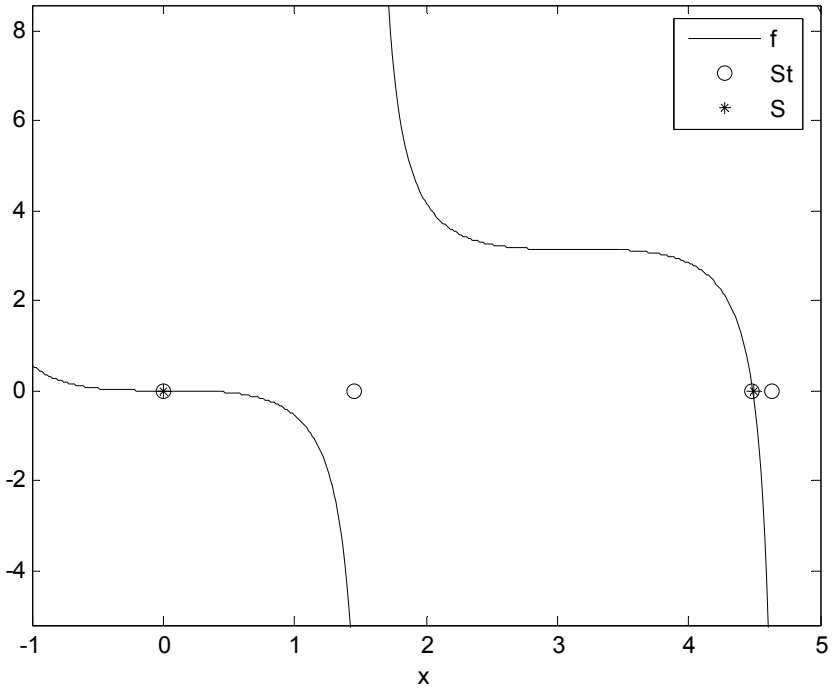
$imax = 30$

$eps = 10^{-7}$ için

$>> S = vnewton('x - tan(x)', St, a, b, df, imax, eps)$

$$S = \begin{bmatrix} -0.00000016977323 \\ 4.49340945790906 \end{bmatrix}, \quad f(S) = 10^{-15} * \begin{bmatrix} 0.00000164112634 \\ -0.88817841970013 \end{bmatrix}$$

elde ederiz.



Şekil 3. $f(x) = x - \tan(x)$ fonksiyonu, St sıfır yeri tahminleri ve hesaplanan S sıfır yerlerinin grafiksel gösterimi

2.4. Kiriş (Sekant) Yöntemi

Bu yöntem bir $f(x) = 0$ denkleminin kökünü, kök civarında seçilen x_0 ve x_1 gibi iki başlangıç değerini kullanmak suretiyle, hesaplamayı amaçlar.

Yöntem verilen x_0 ve x_1 başlangıç değerleri için $(x_0, f(x_0))$ ve $(x_1, f(x_1))$ noktalarından geçen doğrunun x-eksenini kestiği noktayı x_2 olarak alır. Buna göre

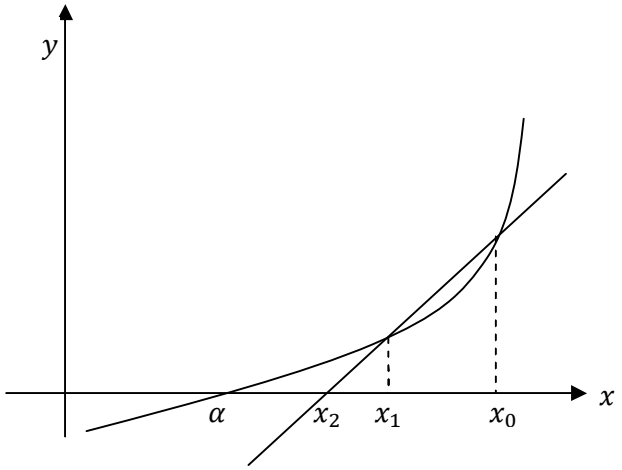
$$0 - f(x_1) = (x_2 - x_1)\frac{\big(f(x_1) - f(x_0)\big)}{(x_1 - x_0)}$$

olup, buradan

$$x_2 = x_1 - f(x_1)\frac{(x_1 - x_0)}{(f(x_1) - f(x_0))}$$

olur. Bu şekilde devam edilerek aşağıdaki genel formül elde edilir.

$$x_{n+1} = x_n - f(x_n)\frac{(x_n - x_{n-1})}{\big(f(x_n) - f(x_{n-1})\big)} \tag{13}$$



Şekil 4. Kiriş yöntemi için grafik çizimi

2.5. Vektör Tabanlı Kiriş Yöntemi

Vektör tabanlı Newton yöntemine benzer şekilde Kiriş yöntemi için vektör tabanlı algoritma aşağıdaki şekilde verilir.

Söz dizimi : $X = vkiriş(f, X0, a, b, imax, eps)$

Girdi: f : Sıfır yerleri belirlenecek fonksiyon

$X0$: Sıfır yerleri için tahmini başlangıç vektörü

a : Aralığın sol uç noktası

b : Aralığın sağ uç noktası

$imax$: Maksimum iterasyon sayısı

eps : Sonuçlandırma kriteri için tolerans

1. $i = 0$; $X = []$; $pert = 0.1$; $X1 = X0 + pert$; Başlangıç indis değerleri ve parametreler
2. $i < imax$ ve $X0 \neq []$ olduğu sürece 3-8 adımları yapınız.
3. $df = \big(f(X1) - f(X0)\big)./(X1 - X0)$; türev için ileri fark yaklaşımı
4. $X0$ vektörünün $df \neq 0$ olan indislerini bul ve if_df0'a ata;
5. $X0 = X0(if\_df0)$, $X1 = X1(if\_df0)$ ve $df = df(if\_df0)$ olarak al;
6. $X2 = X1 - f(X1)./df$ vektörel Kiriş yaklaşımını hesapla;
7. Yakınsayan bileşenleri çözüme ata, yakınsamayanlarla devam et;
   - 7.1. $fark = |X2 - X1|$ vektörünü hesapla;
   - 7.2. $fark > eps$ olan indisleri belirle ve ih vektörüne ata;
   - 7.3. $fark \leq eps$ olan indisleri belirle ve ie vektörüne ata;

7.4. $X2(ie)$ bileşenlerini X çözüm vektörüne ata;

7.5. $X0 = X1(ih)$; $X1 = X2(ih)$;

8. $i = i + 1$;

9. X'de tekrarlanan değerleri ve (a, b) aralığının dışına çıkanları ayıkla ;

Çıktı: X sıfır yerleri

Örnek 2.5.1. $f(x) = x + e^{-50x^2}\cos(x)$ fonksiyonunun $(-1,1)$ aralığındaki sıfır yerlerini bulalım.

$a = -1$

$b = 1$

$\Delta x = 1$

$>> St = fsifirtahmin('x + exp(-50 * x\^2) * cos(x)', a, b, \Delta x)$

$$St = \begin{bmatrix} -0.19366689525009 \\ -0.05834183395834 \\ 0.23191066546043 \end{bmatrix}$$

olur ve $vkiriş$ programını bu başlangıç vektörü için çalıştırırsak

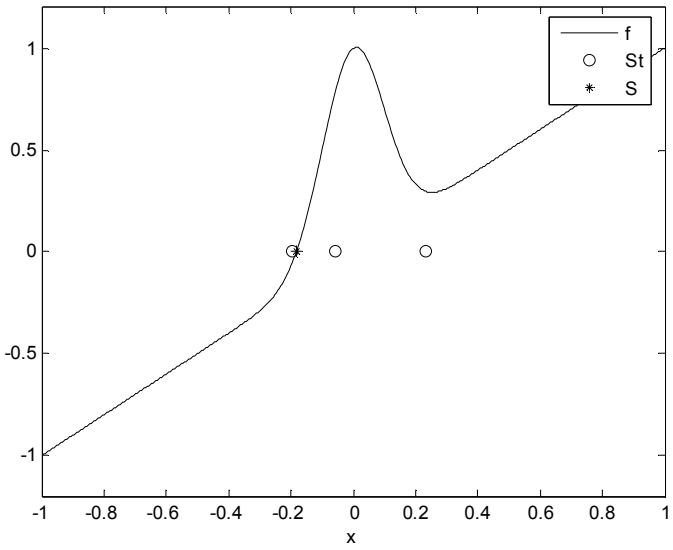
$imax = 30$

$eps = 10\^{} - 7$

$>> S = vkiris('x + exp(-50 * x\^2) * cos(x)', St, a, b, imax, eps)$

$S = [-0.18329133329875]$, $f(S) = [-1.874572719273715e - 011]$

elde ederiz.



Şekil 5. $f(x) = x + e^{-50x^2}\cos(x)$fonksiyonu, St sıfır yeri tahminleri ve $vkiris$ ile elde edilen S sıfır yeri için grafiksel gösterim

Örnek 2.5.2. $f(x) = x^4 - 5x^3 - x^2 + 29x - 26$ fonksiyonunun $(-4,6)$ aralığındaki sıfır yerlerini bulalım.

$a = -4$

$b = 6$

$\Delta x = 1$

$>> St = fsifirtahmin('x\^4 - 5 * x\^3 - x\^2 + 29 * x - 26', a, b, \Delta x)$

$$St = \begin{bmatrix} -3.04520547945205 \\ -1.73295114432390 \\ 1.10685298048301 \\ 1.67503081119583 \\ 2.64960183398532 \\ 3.13574238830102 \\ 3.58095067754768 \end{bmatrix}$$

olur.

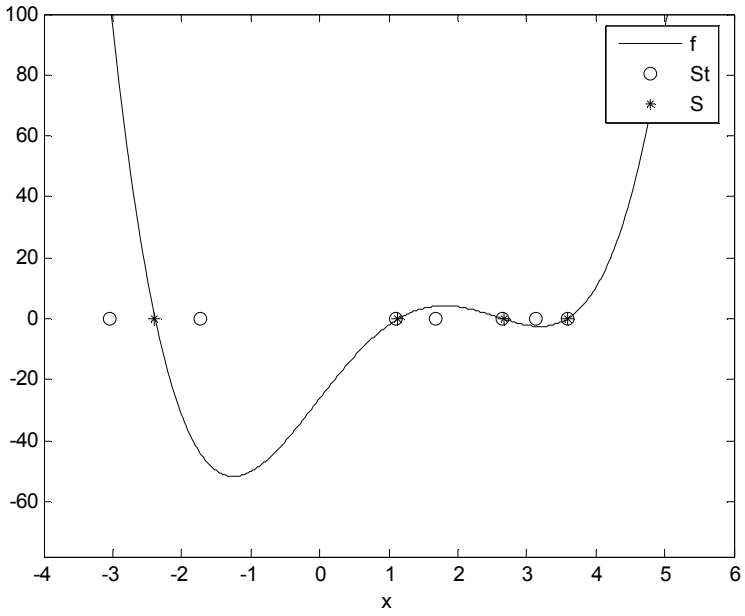
$imax = 30$

$eps = 10\^{} - 7$ için

$>> S = vkiris('x\^4 - 5 * x\^3 - x\^2 + 29 * x - 26', St, a, b, imax, eps)$

$$S = \begin{bmatrix} -2.39126100540448 \\ 1.13684178386669 \\ 2.66321518024408 \\ 3.59120404129371 \end{bmatrix},\ f(S) = 10^{-13} * \begin{bmatrix} 0 \\ 0 \\ -0.07105427357601 \\ -0.28421709430404 \end{bmatrix}$$

elde ederiz.



Şekil 6. $f(x) = x^4 - 5x^3 - x^2 + 29x - 26$ fonksiyonu, St sıfır yeri tahminleri ve $vkiris$ ile elde edilen S sıfır yerleri için grafiksel gösterim

3. VEKTÖR TABANLI SAYISAL OPTİMİZASYON YÖNTEMLERİ

3.1. Dichotomous Yöntemi

Tek değişkenli sürekli bir fonksiyon eğer bir $I = [a, b]$ aralığında tek modlu ise yani, fonksiyon bu aralıkta bir tek minimuma veya maksimuma sahip ise, yöntem fonksiyonun minimumunu, minimum noktayı içeren aralığın uzunluğunu her defasında bir öncekinin yarısı kadar almak suretiyle küçültür ve minimum noktayı bu şekilde belirler.

Öncelikle,

$$c = \frac{a+b}{2}$$

hesaplanır ve daha sonra $\varepsilon > 0$ yeterince küçük bir sabit olmak üzere, eğer

$$f(c-\varepsilon) < f(c+\varepsilon)$$

ise $b = c$ olarak alınır, eğer

$$f(c-\varepsilon) > f(c+\varepsilon)$$

ise $a = c$ olarak alınır, son olarak

$$f(c-\varepsilon) = f(c+\varepsilon)$$

ise c minimumdur. Bu işlem tekrar edilerek her bir adımda aralık ikiye bölünmüş olur yani, k iterasyon sonucunda minimum noktayı içeren aralığın uzunluğu,

$$I_k = \left(\frac{1}{2}\right)^k .(b-a) \tag{14}$$

olur.

3.2. Dik İniş Yöntemi

Dik iniş yöntemi tek değişkenli bir fonksiyonun yerel minimumunu, verilen başlangıç değeri ile oluşturulacak iterasyonda uygun adım uzunlukları ile belirlenen bir yönde ilerleyerek bulmayı amaçlar. Bunun için ilerleme yönü en büyük azalmanın olduğu,

$$-\nabla f = -grad\, f(x) \tag{15}$$

yönünde seçilir. Böylece gradyanı normuna bölerek,

$$P_k = -\frac{\nabla f(x_k)}{\|\nabla F(x_k)\|} \tag{16}$$

en uygun ilerleme yönü seçilmiş olur. Diğer problem adım uzunluğunun belirlenmesidir ki bunun için,

$h(\alpha_k) = f(x_k + \alpha_k P_k)$

tek değişkenli fonksiyonu minimum yapan α_k'lar hesaplanmalıdır. Bu hesaplaması zor işlem yerine önceden belirlenecek makul sabit bir α_k ile her bir adımda,

$f(x_k + \alpha_k P_k) < f(x_k)$

şartı sağlanana kadar α_k'yı yarı oranında azaltarak uygun adım uzunluğu belirlenir. Sonuç olarak iterasyon formülü,

$$x_{k+1} = x_k + \alpha_k P_k \tag{17}$$

şeklinde oluşturulur.

Ancak bu yöntemlerin bir fonksiyona uygulanabilmesi için Dichotomous yönteminde minimum noktayı içeren bir aralığın ve Dik İniş yönteminde minimum noktaya yeterince yakın başlangıç değerinin belirlenmesi gerekir. Bunun için bir aralık üzerinde bütün yerel minimumları tek modlu olarak içerecek alt aralıkları belirleyen minimum tahmin algoritmasını verelim.

3.3. Yerel Minimumlar İçin Tahminler

Kullanıcı tarafından verilen bir (a, b) aralığında tek değişkenli fonksiyonun her bir yerel minimumunu tek modlu olarak içerecek tahmin aralıklarını belirlemek için geliştirilen bir yöntemdir.

Yöntem ikinci bölümde bahsedilen sıfır yeri tahminine benzer şekilde, verilen aralıkta oluşturulacak ağ üzerinde sayısal türevin işaretlerini hesaplayarak işaret değişim noktalarında yerel minimumlar için aralıklar bulmayı amaçlar.

Öncelikle eğer Δx verilmemiş ise aralık uygun bir N için N adet parçaya bölünerek Δx elde edilir. Yani,

$$\Delta x = \frac{b-a}{N} \tag{18}$$

olarak alınır. Sonrasında fonksiyonun bu aralıktaki yerel minimumlarını tahmin etmek için bir ağ oluşturulur. Yerel minimumlarda türev sıfır olduğundan oluşturulacak ağ türevin sıfıra yaklaştığı yerlerde sıklaşacak şekilde seçilmelidir. Bunun için adaptif ağ aşağıdaki şekilde oluşturulabilir.

$$x_{i+1} = x_i + \Delta x \frac{(1 + c|f'(x_i)|)}{(M + c|f'(x_i)|)} \ , \quad x_0 = a \ , M > 1, c > 1 \tag{19}$$

Buradaki türev sayısal türevdir. Daha sonra oluşturulan bu X ağı için $Y = f(X)$ vektörü ile $Y_p = fark\ (Y)$ vektörü oluşturulur ve ayrıca $Y_{p_i} = sgn(Y_p)$ işaret vektörü hesaplanır. Türevin X ağı üzerindeki işaret değişimini belirlemek için

$$test = fark(Y_{p_i})$$

hesaplanır. Sonuç olarak yerel minimumlar türevin negatiften pozitife geçiş noktalarında bulunduğu düşünülerek $test$ vektöründeki sıfırdan büyük bileşenlerin indisleri belirlenip bu indislerin X ağındaki karşılıkları yerel minimumlar için tahmin aralıklarının sol uçları olarak alınır.

3.3.1. Yerel Minimum Tahmin Yönteminin Çözümlemesi

1. f, a, b ve Δx verileri alınır.
2. Eğer Δx verilmemiş ise $\Delta x = (b-a)/N$ alınır.
3. $x_0 = a$, $x_{son} = b$ olmak üzere $X = [x_i]$ ağı oluşturulur.

$$x_{i+1} = x_i + \Delta x \frac{(1 + c|f'(x_i)|)}{(M + c|f'(x_i)|)}$$

4. $Y = f(X)$ hesaplanır.
5. $Y_p = fark(Y)$hesaplanır.
6. Y_p'nin işaretlerinden oluşan Y_{p_i} vektörü oluşturulur.
7. $test = fark(Y_{p_i})$ hesaplanır.
8. $test$'de sıfırdan büyük bileşenlerin indisleri Sti belirlenir.
8. Bu indisler için yerel minimum tahmin aralığı $[X(sti), X(Sti + 2)]$ şeklinde belirlenir.

9. Bulunan tahmin aralıkları geri gönderilir.

3.3.2. Yerel Minimumlar İçin Tahmin Algoritması

Söz dizimi : $St = fmintahmin(f, a, b, \Delta x)$

Girdi: f : Yerel minimumları belirlenecek fonksiyon

a : Aralığın sol uç noktası

b : Aralığın sağ uç noktası

Δx : En büyük adım uzunluğu

1. $i = 1;\ N = 20; X = a;\ M = 100; c = 5;$ Başlangıç indis değerleri ve parametreler
2. Eğer Δx verilmemiş ise $\Delta x = (b - a)/N;$
3. $h = 0.01\Delta x;$
4. $X(son) < b$ olduğu sürece
   - 4.1. $dfxi = (f(X(i) + h) - f(X(i)))/h;$
   - 4.2. $x1 = X(i) + \Delta x \frac{(1+c|dfxi|)}{(M+c|dfxi|)}$; Adaptif ağ formülünü uygula
   - 4.3. $X = [X, x1];$
   - 4.4. $i = i + 1;$
5. Eğer $X(son) < b$ ise $X = [X, b]$ değilse $X(son) = b$;
6. $Y = f(X);$
7. $Yp = fark(Y);$
8. $Ypi = işaret(Yp);$ Yp'nin işaret vektörü
9. $test = fark(Ypi);$ işaret değişimi
10. $Sti = indis(test > 0);$ işaretin negatiften pozitife geçtiği indisler
11. $Sta = [X(Sti),\ X(Sti + 2)]$;

Çıktı: Sta yerel minimumlar için tahmin aralıkları;

Örnek 3.3.1. $f(x) = \sin(x)$ fonksiyonunun $(-5,5)$ aralığındaki her bir yerel minimumu için birer tahmin aralığı belirleyelim. Bunun için $fmintahmin$ programını çalıştıralım.

$a = -5$

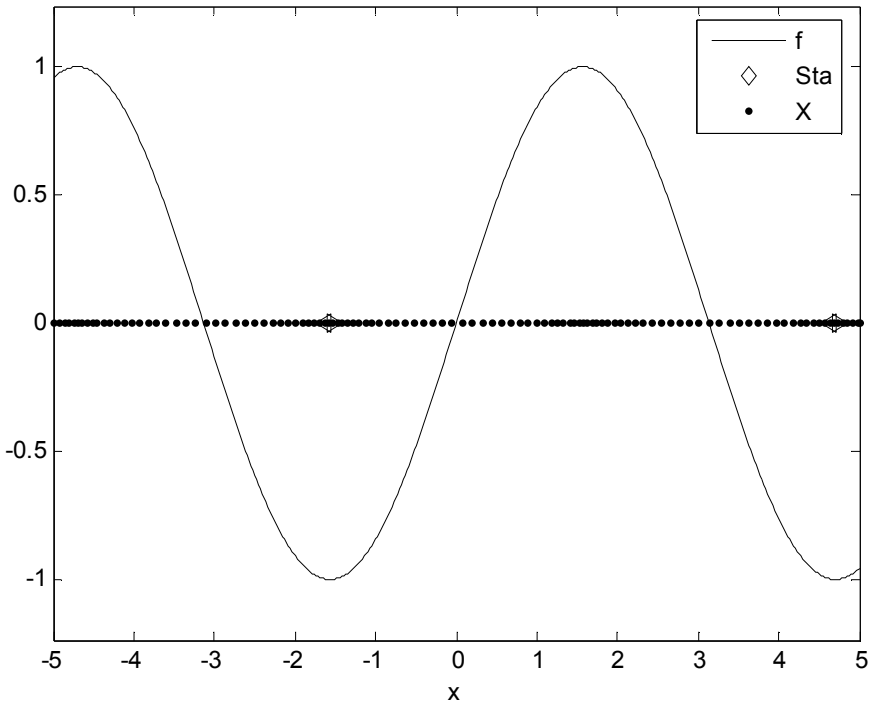
$b = 5$

$\Delta x = 0.5$

>>$Sta = fmintahmin('\sin(x)', a, b, \Delta x)$

$$Sta = \begin{bmatrix} -1.63259662250542 & -1.52696290508963 \\ 4.63321493177427 & 4.74178717798801 \end{bmatrix}$$

elde ederiz.



Şekil 7. $f(x) = \sin(x)$ fonksiyonu, Sta yerel minimum tahmin aralıkları ve oluşturulan X ağı için grafik

3.4. Vektör Tabanlı Dichotomous Yöntemi

Dichotomous yöntemi için bir aralık içersindeki vektör değerli tahmin aralıklarını kullanarak bütün yerel minimumları hesaplayacak şekilde vektör tabanlı algoritma aşağıdaki gibi verilebilir.

Söz dizimi : $S = vdichotomous(f, Sta, eps)$

Girdi: f : Yerel minimumları belirlenecek fonksiyon

Sta : $fmintahmin$ ile belirlenen vektörel aralık

eps : Sonuçlandırma kriteri için tolerans

1. $S = []$;
2. $Sta(:,2) - Sta(:,1) > eps$ olduğu sürece 3-8 adımları yapınız;
3. $c = (Sta(:,1) + Sta(:,2))/2$;
4.

4.1. $j1 = indis(f(c - eps) < f(c + eps))$;

4.2. $j2 = indis(f(c - eps) > f(c + eps))$;

4.3. $j3 = indis(f(c - eps) = f(c + eps))$;

5. $Sta(j1,2) = c(j1)$; $Sta(j2,1) = c(j2)$; yeni aralıkları belirle

6. $S = [S; c(j3)]$; $Sta(j3, :) = []$; minimumları çözüme at ve Sta'dan çıkar

7.

7.1. $yi = indis((Sta(:,2) - Sta(:,1)) < eps)$; yakınsayan indisleri yi'ye ata

7.2. $c = (Sta(yi, 1) + Sta(yi, 2))/2$; $S = [S; c]$; yakınsayan indisleri c'ye ata ve çözüme ilave et

8. $Sta(yi, :) = []$;yakınsayan bileşenleri Sta'dan çıkar

Çıktı: Yerel minimumlardan oluşan S vektörü

Örnek 3.4.1. $f(x) = -sin(x) - sin(3x)/3$ fonksiyonun $(-10,10)$ aralığındaki bütün yerel minimumlarını belirleyelim. Bunun için $fmintahmin$ ve $vdichotomous$ programlarını çalıştıralım.

$a = -10$

$b = 10$

$\Delta x = 0.5$

$>> Sta = fmintahmin('-sin(x) - sin(3 * x)/3', a, b, \Delta x)$

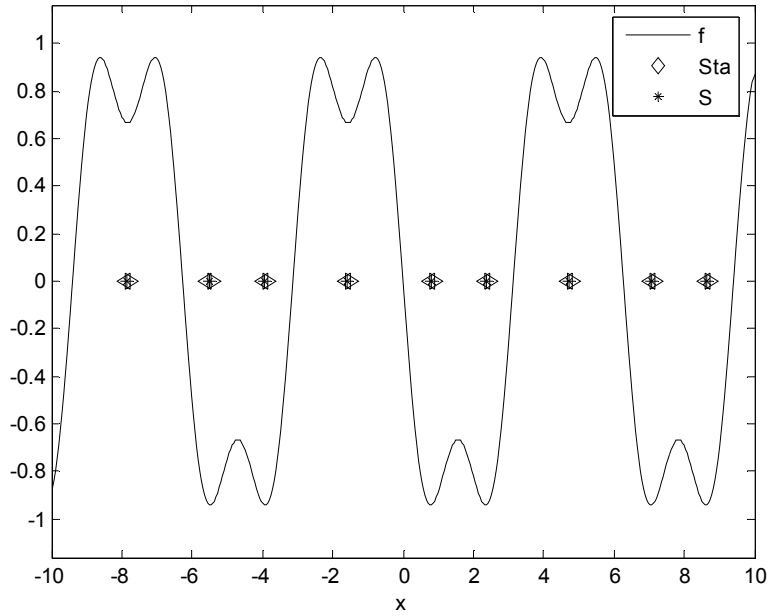
$$Sta = \begin{bmatrix} -7.94303116758584 & -7.82433596242835 \\ -5.56068450074036 & -5.44368372911006 \\ -4.02476691688834 & -3.89804729232138 \\ -1.62845808066958 & -1.51799927322853 \\ 0.75172368267324 & 0.86630164800908 \\ 2.29301271273439 & 2.40785457727285 \\ 4.68060841215769 & 4.79041915436510 \\ 6.99538097507601 & 7.11435198757897 \\ 8.60369759870584 & 8.71808893679634 \end{bmatrix}$$

$eps = 10\text{^} - 8$

$>> S = vdichotomous('-sin(x) - sin(3 * x)/3', Sta, eps)$

$$S = \begin{bmatrix} -7.85398163437041 \\ -5.49778715114936 \\ -3.92699082019478 \\ -1.57079632328927 \\ 0.78539816524496 \\ 2.35619448835186 \\ 4.71238898069308 \\ 7.06858346674573 \\ 8.63937979797795 \end{bmatrix}$$

elde ederiz.



Şekil 8. $f(x) = -sin(x) - sin(3x)/3$ fonksiyonu için yerel minimumlar

3.5. Tek Değişkenli Fonksiyonların Minimizasyonu İçin Vektör Tabanlı Dik İniş Yöntemi

Söz dizimi : $X = vdikinis(f, X0, a, b, df, imax, eps)$

Girdi: f : Yerel minimumları belirlenecek fonksiyon

$X0$: Yerel minimumlar için tahmini başlangıç vektörü

a : Aralığın sol uç nokrası

b : Aralığın sağ uç noktası

df : f'nin türevi

$imax$: Maksimum iterasyon sayısı

eps : Sonuçlandırma kriteri için tolerans

1. $i = 0$; $X = []$; $\alpha = 1$; başlangıç indis değerleri ve parametreler
2. $X0$ ile aynı boyutta α'lardan oluşan A vektörünü tanımla
3. $i < imax$ ve $X0 \neq []$ olduğu sürece 4-11 adımları yapınız.
4.
   4.1. $X0$ vektörünün $|df(X0)| < eps$ olan indislerini belirle ve $df0$'a ata;
   4.2. $X0$ vektörünün $|df(X0)| \geq eps$ olan indislerini bul ve $df1$'e ata;
5. $X = [X; X0(df0)]$; türev yeterince küçük ise çözüme at

6. $X0 = X0(df1)$; $A = A(df1)$; türevin yeterince küçük olmadığı noktalarla devam et

7. $X1 = X0 - A.* df(X0)./abs(df(X0))$; Dik İniş formülünü vektörel formda uygula

8. $ay = indis(f(X1) > f(X0))$; $f(X1) > f(X0)$ olan indisleri belirle

9. $ay \neq$ [] olduğu sürece

   9.1. $A(ay) = A(ay)./2$;

   9.2. $X1 = X0 - A.* df(X0)./abs(df(X0))$;

   9.3. $ay = indis(f(X1) > f(X0))$;

10 $Ui = indis(A \geq eps/2)$; $X0 = X1(Ui)$; uygun ilerleme olan noktalarla devam et

11. $i = i + 1$;

12. X'de tekrarlanan değerleri ve (a, b) aralığının dışına çıkanları ayıkla;

Çıktı: Yerel minimumlardan oluşan X vektörü

Örnek 3.5.1. $f(x) = sin(x) - xe^{\sin(-x)}$ fonksiyonun $(-1,20)$ aralığındaki bütün yerel minimumlarını belirleyelim. Bunun için $fmintahmin$ ve $vdikinis$ programlarını peşi sıra çalıştıralım.

$a = -1$

$b = 20$

$\Delta x = 1$

$>> Sta = fmintahmin('sin(x) - x * exp(sin(-x))', a, b, \Delta x)$

$$Sta = \begin{bmatrix} -0.01285669498007 & 0.00821549060872 \\ 4.88413271067366 & 4.91810958744379 \\ 11.06662858407007 & 11.12769895552945 \\ 16.94981325560356 & 17.58417134701670 \end{bmatrix}$$

$df = cos(x) - exp(-sin(x)) + x * cos(x) * exp(-sin(x))$

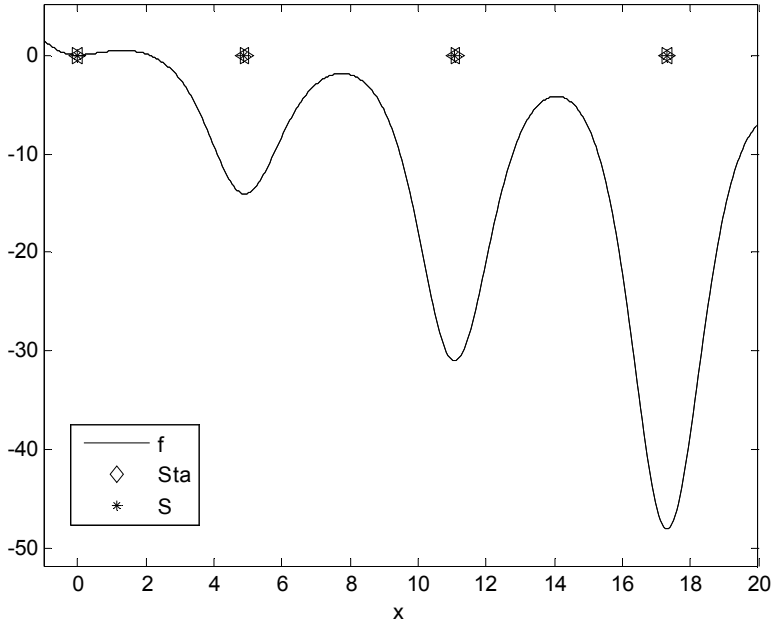
$imax = 50$

$eps = 10\text{^} - 6$

$>> S = vdikinis('sin(x) - x * exp(sin(-x))', Sta(:,1), a, b, df, imax, eps)$

$$S = \begin{bmatrix} -0.00000021152060 \\ 4.90302070734724 \\ 11.08300442378031 \\ 17.33527491384292 \end{bmatrix}$$

elde ederiz.



Şekil 9. $f(x) = sin(x) - xe^{\sin(-x)}$ fonksiyonu için yerel minimumlar

Ayrıca $-f(x)$ için yöntemi çalıştırarak $f(x)$ fonksiyonun yerel maksimumlarını da bulabiliriz.

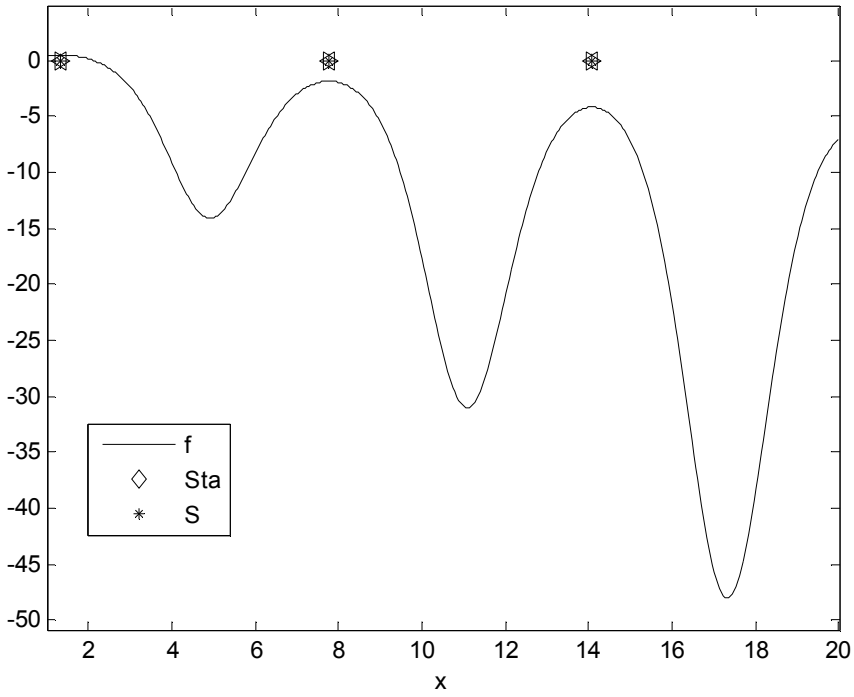
$>> Sta = fmintahmin('-sin(x)+x*exp(sin(-x))', a, b, \Delta x)$

$$Sta = \begin{bmatrix} 1.30707301318989 & 1.32772295262440 \\ 7.74767320781561 & 7.76975277647545 \\ 14.06486389726280 & 14.08861404808556 \end{bmatrix}$$

$$df = -cos(x) + exp(-sin(x)) - x * cos(x) * exp(-sin(x))$$

$$S = vdikinis('-sin(x)+x*exp(sin(-x))', Sta(:,1), a, b, df, imax, eps)$$

$$S = \begin{bmatrix} 1.31454791248188 \\ 7.75827091365667 \\ 7.75827091365667 \end{bmatrix}$$



Şekil 10. $f(x) = sin(x) - xe^{\sin(-x)}$ fonksiyonu için yerel maksimumlar

Örnek 3.5.2. $f(x) = \tan(x)^2$ fonksiyonun $(0,10)$ aralığındaki bütün yerel minimumlarını belirleyelim.

$a = 0$

$b = 10$

$\Delta x = 1$ için

$>> Sta = fmintahmin('tan(x)^2', a, b, \Delta x)$

$$Sta = \begin{bmatrix} 3.13175492824998 & 3.15279184963898 \\ 6.27604708149377 & 6.29705744887538 \\ 9.41077717892418 & 9.43185462491131 \end{bmatrix}$$

$df = cos(x) - exp(-sin(x)) + x * cos(x) * exp(-sin(x))$

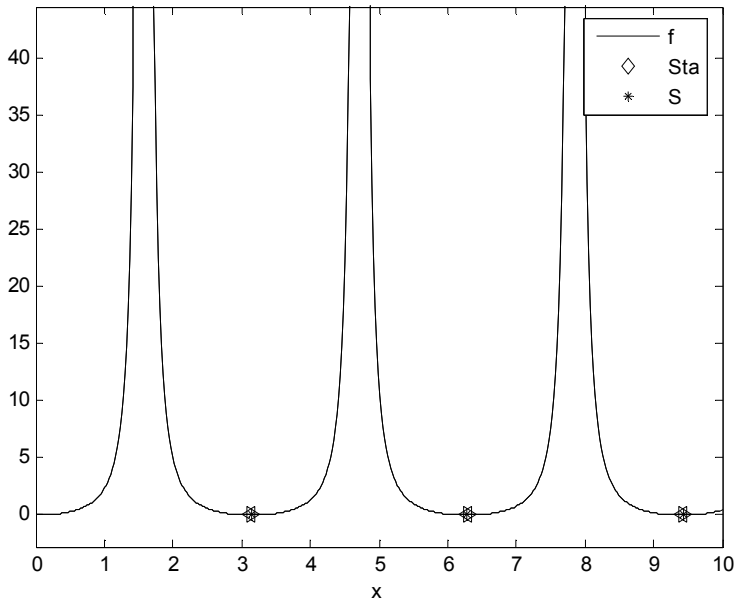
$imax = 30$

$eps = 10\text{^} - 6$

$>> S = vdikinis('tan(x)^2', Sta(:,1), a, b, df, imax, eps)$

$$S = \begin{bmatrix} 3.14159303249803 \\ 6.28318533375207 \\ 9.42477807156334 \end{bmatrix}$$

elde ederiz.



Şekil 11. $f(x) = \tan(x)^2$ fonksiyonu için hesaplanan yerel minimumlar

4. LİNER OLMAYAN DENKLEM SİSTEMLERİNİN SAYISAL ÇÖZÜMLERİ

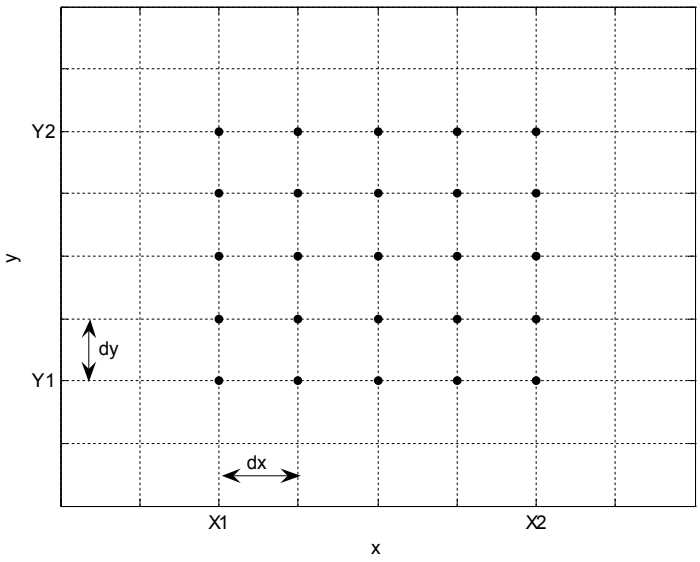
4.1. Denklem Sistemleri İçin Çözüm Tahminleri

Kullanıcı tarafından belirlenen bir bölgede, iki değişkenli lineer olmayan denklem sisteminin olabilecek bütün reel ayrık çözümleri için birer tahmin belirlemek amacıyla geliştirilen bir yöntemdir.

Kullanıcı $f_1(x,y) = 0$, $f_2(x,y) = 0$ denklemlerini, $X \times Y$ bölgesi için $X = (X_1, X_2)$ $Y = (Y_1, Y_2)$ aralıklarını ve eğer ağın hassasiyetini belirlemek isterse Δx ve Δy artım miktarlarını verir. Yöntem bu bölgede var olabilecek bütün reel ayrık çözümler için birer tahmin bulmayı amaçlar.

Öncelikle eğer Δx ve Δy artım miktarları verilmemiş ise ağ için artım miktarları X ve Y aralıkları sırasıyla uygun N ve M eşit parçalara bölünerek hesaplanır. Sonrasında bu artım miktarları ile $X \times Y$ bölgesinde Ω ağı aşağıdaki gibi oluşturulur.

$$\Omega_{ij} = \{(x_i, y_j) \mid x_i \in [X_1 : \Delta x : X_2], y_j \in [Y_1 : \Delta y : Y_2]\} \tag{20}$$



Şekil 12. Oluşturulan Ω ağı için grafik

Daha sonra

$$F(x,y) = f_1(x,y)^2 + f_2(x,y)^2$$

fonksiyonu tanımlanır ve alınan Ω ağındaki değerleri hesaplanır. Denklem sisteminin çözümünde F'nin gradyanı $\nabla F = 0$ olduğundan sistem için çözüm tahminleri olarak bu noktalar seçilebilir. Bu noktaları, alınan ağdan sayısal olarak hesaplamak mümkündür. Bunun için Ω ağında $Z = F(\Omega)$ matrisi hesaplanır ve bu matrisin satırlara ve sütunlara göre fark matrisleri hesaplanır ve ardından fark matrislerinin işaret değişim noktaları belirlenir. Bu işaret değişim noktaları aranan çözüm tahminleridir.

4.1.1. Tahmin Yönteminin Çözümlemesi

1. . $f_1, f_2, X = (X_1, X_2),\ Y = (Y_1, Y_2),\ \Delta x$ ve Δy verileri alınır.
2. Eğer Δx veya Δy verilmemiş ise artım miktarları belirlenir $(\Delta x = (X_2 - X_1)/N$, $\Delta y = (Y_2 - Y_1)/M$, tipik olarak $N = M = 100$ alınabilir).
3. $F(x,y) = f_1(x,y)^2 + f_2(x,y)^2$ fonksiyonu tanımlanır.
4. $\Omega_{ij} = \{(x_i, y_j) |\ x_i \in [X_1 : \Delta x : X_2], y_j \in [Y_1 : \Delta y : Y_2]\}$ ağı oluşturulur.
5. $Z = F(\Omega)$ matrisi hesaplanır.
6. Sayısal türevler hesaplanır.
   - 6.1. Z' nin sütunlara göre fark matrisi dFx hesaplanır.
   - 6.2. Z' nin satırlara göre fark matrisi dFy hesaplanır.
7. Türevlerin işaretleri belirlenir.
   - 7.1. dFx'in işaretlerinden oluşan matris dFxi hesaplanır.
   - 7.2 dFy'in işaretlerinden oluşan matris dFyi hesaplanır.
8. İşaret değişimleri belirlenir.
   - 8.1. dFxi' nin sütunlara göre fark matrisi d$Fxid$ hesaplanır.
   - 8.2. dFyi' nin satırlara göre fark matrisi d$Fyid$ hesaplanır.
10. $test =$ d$Fxid.*$ d$Fyid$ hesaplanır.
11. $test$'de sıfırdan farklı değerlerin indisleri belirlenir.
12. Bulunan indislerin Ω ağındaki karşılıkları çözüm tahminleri olarak alınır.
13. Bulunan çözüm tahminleri geri gönderilir.

4.1.2. Denklem Sistemi Çözüm Tahminleri İçin Algoritma

Söz dizimi : $St = fsifirtahmindenksis(f1, f2, X, Y, \Delta x, \Delta y)$

Girdi: $f1$, $f2$: Denklem sistemi için fonksiyonlar

X : Bölge için $X = (X1, X2)$ aralığı

Y : Bölge için $Y = (Y1, Y2)$ aralığı

Δx : x-ekseni için artım miktarı

Δy : y-ekseni için artım miktarı

1. $N = 100$; $M = 100$;

3. Eğer Δy verilmemiş ise $\Delta y = \Delta x$;

4. Eğer Δx ve Δy verilmemiş ise $\Delta x = (X2 - X1)/N$, $\Delta y = (Y2 - Y1)/M$ al;

5. $F(x, y) = f1(x, y)^2 + f2(x, y)^2$;

6. $\Omega_{ij} = \{(x_i, y_j) | \, x_i \in [X_1 : \Delta x : X_2], y_j \in [Y_1 : \Delta y : Y_2]\}$

7. $Z = F(\Omega)$;

8. Sayısal türevleri hesapla

8.1. Z' nin sütunlara göre fark matrisini hesapla ve dFx'e ata;

8.2. Z' nin satırlara göre fark matrisini hesapla ve dFy'ye ata;

9. Türevlerin işaretlerini belirle

9.1. dFx'in işaretlerini belirle ve dFxi'ye ata;

9.2 dFy'in işaretleri belirle ve dFyi'ye ata;

10. İşaret değişimlerini belirle

10.1. dFxi' nin sütunlara göre fark matrisini d$Fxid$'ye ata;

10.2. dFyi' nin satırlara göre fark matrisini d$Fyid$'ye ata;

11. $test =$ d$Fxid.*$ d$Fyid$;

12. $[i, j] = indis(test \neq 0)$;

13. $St = [x(i), y(j)]$; bulunan indislerin ağdaki karşılığını St'ye ata.

Çıktı: St sıfır yeri tahminleri

Örnek 4.1.1. $x^2 + y^2 = 4$, $x^2 + y^2 - 2x - 3y + 1 = 0$ denklemleri ile verilen sistemin $(-2,3) \times (-2,3)$ bölgesindeki çözümleri için tahminler belirleyelim. Bunun için Ekte verilen $fsifirtahmindenksis$ programını çalıştırırsak

$>> f1 = 'x\^2 + y\^2 - 4'$;

$>> f2 = 'x\^2 + y\^2 - 2 * x - 3 * y + 1'$;

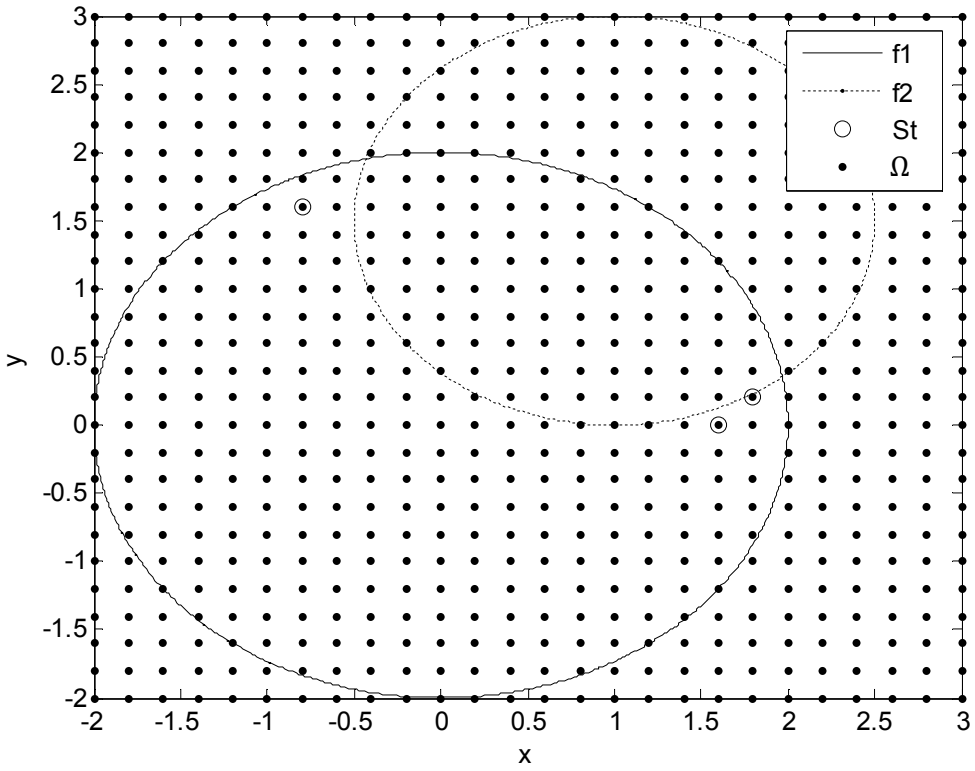
$>>X = [-2,3]$; $Y = [-2,3]$;

$>>\Delta x = 0.2$; $\Delta y = 0.2$;

$>>St = fsifirtahmindenksis(f1, f2, X, Y, \Delta x, \Delta y)$

$$St = \begin{bmatrix} -0.8 & 1.6 \\ 1.6 & 0 \\ 1.8 & 0.2 \end{bmatrix}$$

elde ederiz.



Şekil 13. $x^2 + y^2 = 4$, $x^2 + y^2 - 2x - 3y + 1 = 0$ denklemleri, St çözüm tahminleri ve oluşturulan Ω ağı

4.2. Denklem Sisteminin Çözümü İçin Vektör Tabanlı Dik İniş Yöntemi

Söz dizimi : $S = vdikinisdenk(f11, f22, St, df1, df2, eps, imax)$

Girdi: $f1$, $f2$: Denklem sistemi için fonksiyonlar

St : Çözüm için tahminler

$df1$: $f1$'in gradyanı

$df2$: $f2$'nin gradyanı

eps : Sonuçlandırma kriteri için tolerans

$imax$: Maksimum iterasyon sayısı

1. $i = 0$; $S = [\,]$; $\alpha = 1$; $eps = eps^2$; başlangıç indis değerleri ve parametreler

2. St ile aynı boyutta α'lardan oluşan A vektörünü tanımla

3. $F(x,y) = f1(x,y)^2 + f2(x,y)^2$ fonksiyonunu tanımla;

4. $i < imax$ ve $St \neq [\,]$ olduğu sürece 5-13 adımları yapınız.

5.

5.1. F'nin gradyanı GF'yi ve $Fd = F(St)$'yi hesapla;

5.2. $G0 = indis(\|GF\| < eps)$;gradyanın normu eps'den küçük olan indisleri belirle

5.3. $F0 = indis(|Fd| < eps)$; F' de yeterince küçük değerler alan noktaları belirle

5.4. $gf0 = G0 \cup F0$;

6. $S = [S; St(F0,:)]$; F'nin değeri yeterince küçük olan noktaları çözüme at

7. $c = gfo'$; gfo'ın tümleyenini hesapla

8. $St = St(c,:)$; $A = A(c,:)$; $Fd = Fd(c,:)$; uygun noktalarla devam et

9. $S1 = St - A.* GF./\|GF\|$; Dik İniş formülünü vektörel formda uygula

10. $ay = indis(F(S1) \geq F(St))$; $F(S1) \geq F(St)$ olan indisleri belirle

11. $ay \neq [\,]$ olduğu sürece

11.1. $A(ay,:) = A(ay,:)./2$;

11.2. $S1 = St - A.* GF./\|GF\|$;

11.3. $ay = indis(F(S1) \geq F(St))$;

12 $Ui = indis(A(:,1) \geq eps/2)$; $St = St(Ui,:)$; uygun ilerleme olan noktalarla devam et

13. $i = i + 1$;

14. S'de tekrarlanan bileşenleri ayıkla

Çıktı: S denklem sisteminin çözümleri

Örnek 4.2.1. $x^2 - y - 0.2 = 0$, $y^2 - x - 0.3 = 0$ denklemleri ile verilen sistemin $(-2,2) \times (-2,2)$ bölgesindeki çözümlerini hesaplayalım. Bunun için Ekte verilen $fsifirtahmindenksis$ ve $vdikinisdenk$ programlarını sırası ile çalıştıralım.

>>$f1 = 'x\text{^}2 - y - 0.2'$;

>>$f2 = 'y\text{^}2 - x - 0.3'$;

>>$X = [-2,2]$; $Y = [-2,2]$;

>>$\Delta x = 0.05$; $\Delta y = 0.05$;

için

>>$St = fsifirtahmindenksis(f1, f2, X, Y, \Delta x, \Delta y)$

$$St = \begin{bmatrix} -0.35 & -0.10 \\ -0.30 & -0.15 \\ 0.55 & 0.60 \\ 0.60 & 0.65 \\ 0.95 & 1.00 \\ 1.00 & 1.05 \end{bmatrix}$$

$>> df1 = [\,2x, -1];\ df2 = [\,-1, 2y];$

$>>\ eps = 10\text{^} - 8;\ imax = 100;$

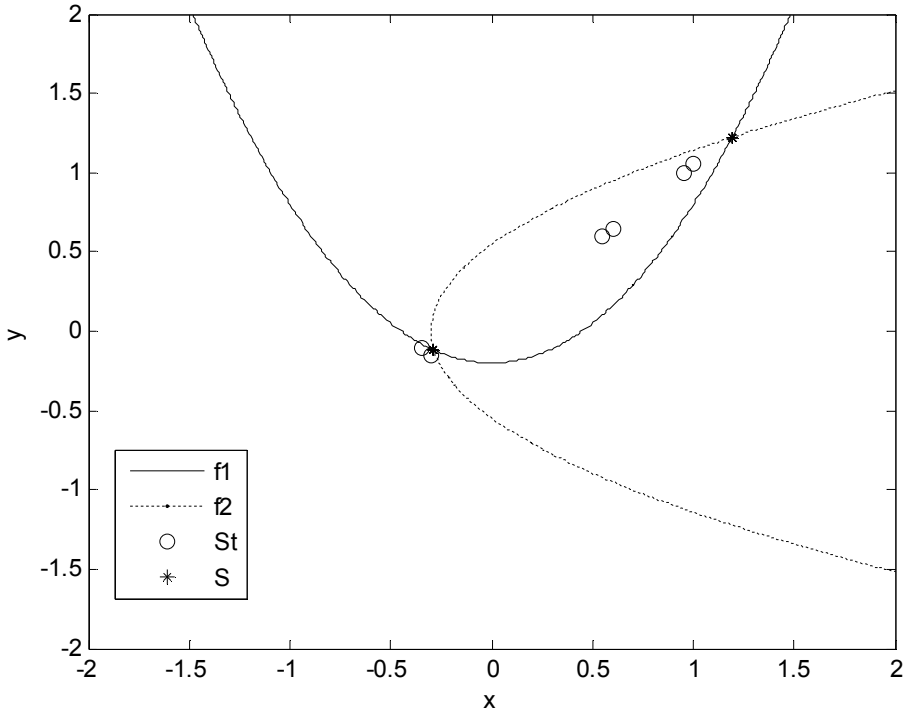
$>>\ S = vdikinisdenk(f1, f2, St, df1, df2, eps, imax)$

$$S = \begin{bmatrix} -0.2860322 & -0.1181856 \\ 1.1923091 & 1.2216010 \end{bmatrix}$$

$$f1(S) = 10^{-7} * \begin{bmatrix} 0.1943683 \\ -0.1005719 \end{bmatrix}$$

$$f2(S) = 10^{-7} * \begin{bmatrix} 0.3604736 \\ -0.9679900 \end{bmatrix}$$

sonuçlarını elde ederiz.



Şekil 14. $x^2 - y - 0.2 = 0$, $y^2 - x - 0.3 = 0$ denklemleri, St çözüm tahminleri ve S çözüm

Örnek 4.2.2. $x^3 - 3x^2 + 4x - y = 0$, $y^2 - x - 2 = 0$ denklemleri ile verilen sistemin $(-2,2) \times (-2,3)$ bölgesindeki çözümlerini hesaplayalım.

$>>f1 = 'x\text{^}3 - 3 * x\text{^}2 + 4 * x - y';$

$>>f2 = 'y\text{^}2 - x - 2'\ ;$

$>>X = [-2,2]\ ; Y = [-2,3]\ ;$

$>>\Delta x = 0.1; \Delta y = 0.1;$

$>>St = fsifirtahmindenksis(f1, f2, X, Y, \Delta x, \Delta y)$

$$St = \begin{bmatrix} 0.4 & -1.5 \\ 0.2 & 1.3 \\ 0.3 & 1.4 \end{bmatrix}$$

$>> eps = 10\text{^} - 8; imax = 100;$

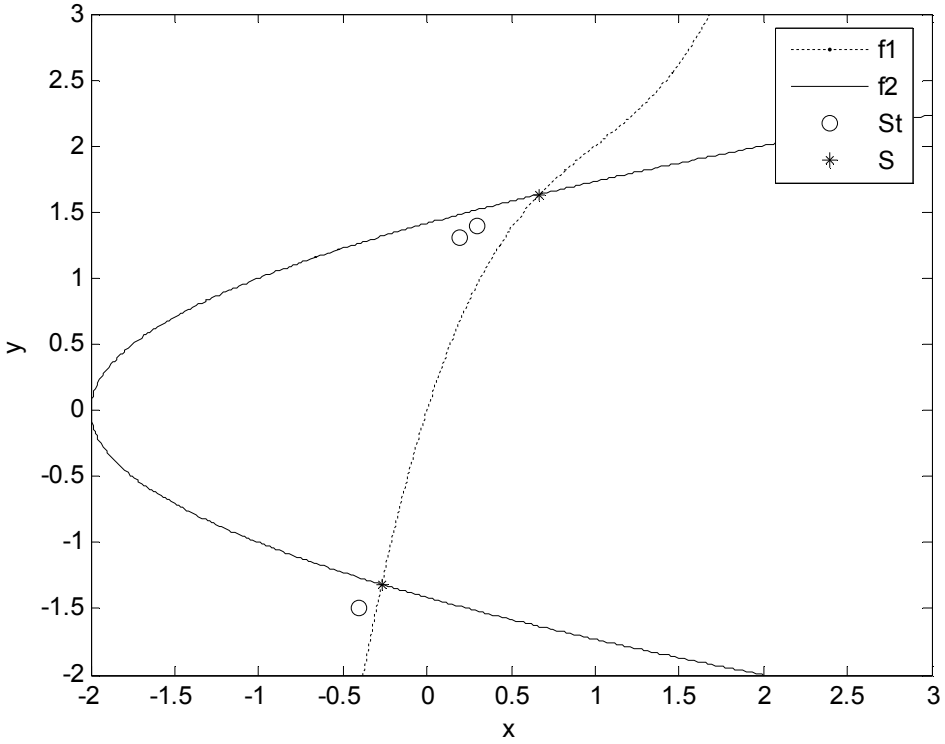
$>>df1 = [\ 3 * x\text{^}2 - 6 * x + 4,\ -1]; df2 = [\ -1, 2 * y];$

$>> S = vdikinisdenk(f1, f2, St, df1, df2, eps, imax)$

$$S = \begin{bmatrix} -0.2695032 & -1.3154835 \\ 0.6699515 & 1.6339986 \end{bmatrix}$$

$$f1(S) = 10^{-6} * \begin{bmatrix} 0.2259356 \\ 0.0527211 \end{bmatrix}$$

$$f2(S) = 10^{-7} * \begin{bmatrix} 0.3877225 \\ -0.7519803 \end{bmatrix}$$



Şekil 15. $x^3 - 3x^2 + 4x - y = 0,\ y^2 - x - 2 = 0$ denklemleri, St çözüm tahminleri ve S çözüm

Örnek 4.2.3. $y+\sin(x)=0,\ y+\cos(x)=0$ denklemleri ile verilen sistemin $(0,15)\times(-2,2)$ bölgesindeki çözümlerini hesaplayalım. Bunun için $fsifirtahmindenksis$ ve $vdikinisdenk$ programlarını çalıştırırsak

$>>f1='y+\sin(x)';$

$>>f2='y+cos(x)';$

$>>X=[0,15]\ ;Y=[-2,2]\ ;$

$>>\Delta x=0.1;\ \Delta y=0.1;$

$>>St=fsifirtahmindenksis(f1,f2,X,Y,\Delta x,\Delta y)$

$$St=\begin{bmatrix}0.7 & -0.8\\ 2.5 & 0\\ 3.8 & 0.6\\ 7.0 & -0.8\\ 8.8 & 0\\ 10.1 & 0.6\\ 11.7 & 0\\ 13.3 & -0.8\end{bmatrix}$$

$>>eps=10\hat{}-8;\ imax=100;$

$>>df1=[\,cos(x),1];\ df2=[\,-sin(x),1];$

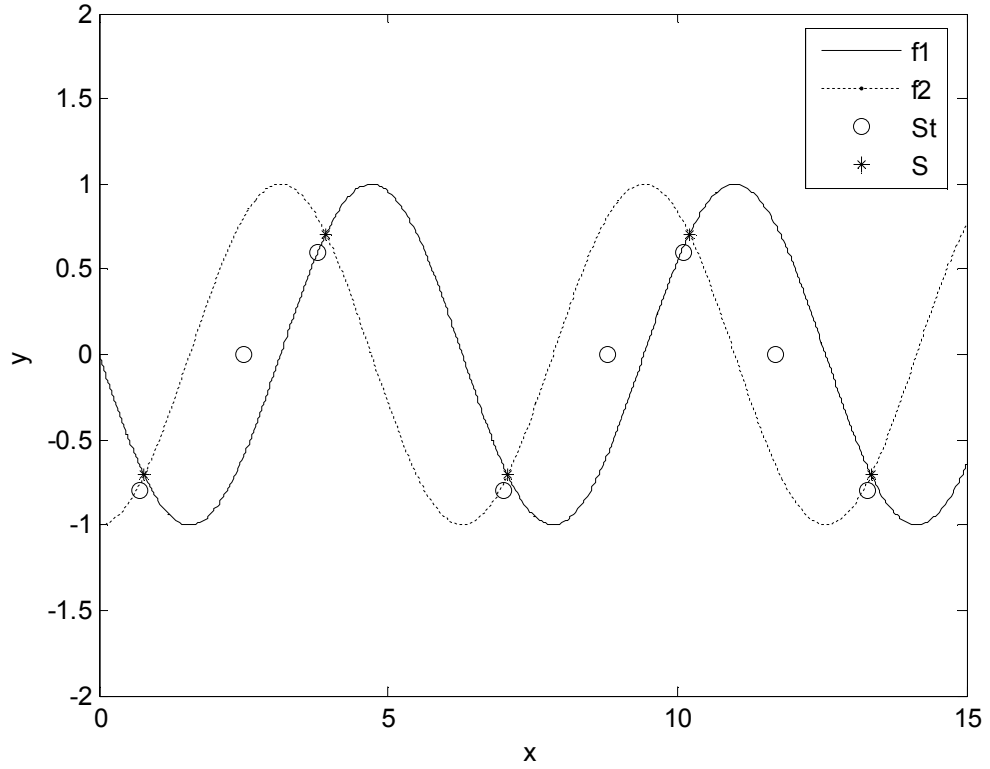
$>>S=vdikinisdenk(f1,f2,St,df1,df2,eps,imax)$

$$S=\begin{bmatrix}0.7853982 & -0.7071068\\ 3.9269908 & 0.7071068\\ 7.0685835 & -0.7071068\\ 10.2101761 & 0.7071068\\ 13.3517688 & -0.7071068\end{bmatrix}$$

$$f1(S)=10^{-7}*\begin{bmatrix}0.0706845\\ 0.3082524\\ 0.0199172\\ 0.3590198\\ -0.0308500\end{bmatrix}$$

$$f2(S)=10^{-7}*\begin{bmatrix}-0.4469536\\ 0.0680165\\ -0.3961863\\ 0.0172492\\ -0.3454189\end{bmatrix}$$

sonuçlarını elde ederiz.



Şekil 16. $y + \sin(x) = 0,\ y + \cos(x) = 0$ denklemleri, St çözüm tahminleri ve S çözüm için grafik

Örnek 4.2.4. $\frac{dx}{dt} = 2\sin(x) + \sin(y),\ \frac{dy}{dt} = \sin(x) + 2\sin(y)$ diferensiyel denkleminin $(-4,4) \times (-4,4)$ bölgesindeki kritik noktalarını belirleyelim. Bunun için $fsifirtahmindenksis$ ve $vdikinisdenk$ programlarını çalıştırırsak

>>$f1 = '2 * sin(x) + sin(y)'$;

>>$f2 = 'sin(x) + 2 * sin(y)'$;

>>$X = [0,15]$; $Y = [-2,2]$;

>>$\Delta x = 0.1$; $\Delta y = 0.1$;

>> $St = fsifirtahmindenksis(f1, f2, X, Y, \Delta x, \Delta y)$;

>> $eps = 10\text{^} - 9$; $imax = 100$;

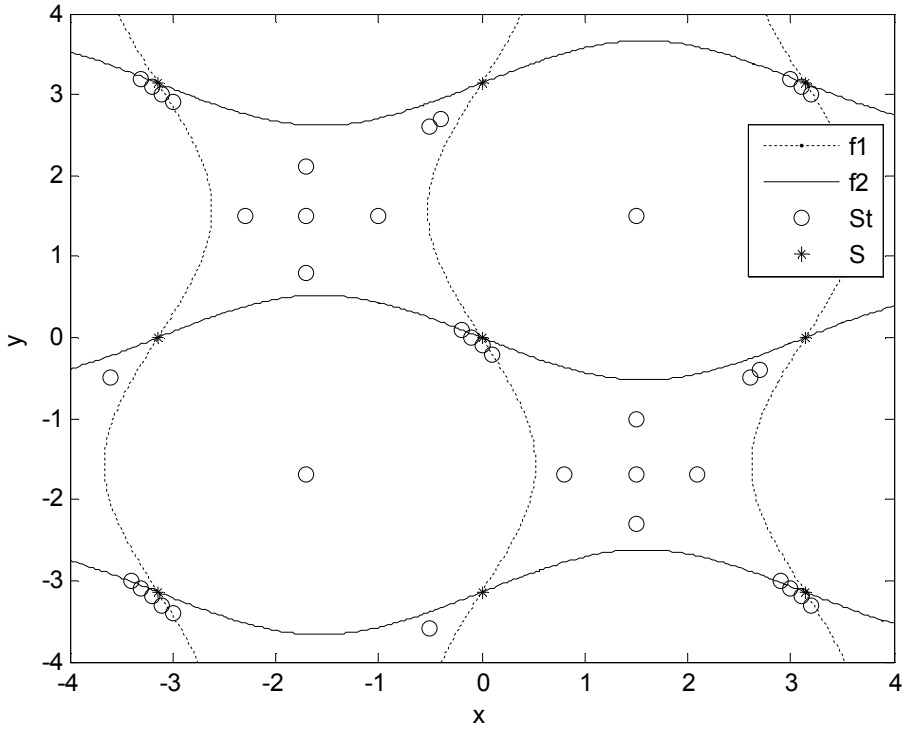
>>$df1 = [\ 2 * cos(x),\ \ cos(y)]$; $df2 = [cos(x), 2 * cos(y)]$;

>> $S = vdikinisdenk(f1, f2, St, df1, df2, eps, imax)$

$$S = \begin{bmatrix} -3.14159265 & -3.14159265 \\ -3.14159265 & 0 \\ -3.14159265 & 3.14159265 \\ 0 & -3.14159265 \\ 0 & 0 \\ 0 & 3.14159265 \\ 3.14159265 & -3.14159265 \\ 3.14159265 & 0 \\ 3.14159265 & 3.14159265 \end{bmatrix}$$

$$f1(S) = 10^{-7} * \begin{bmatrix} -0.1076937 \\ -0.0717958 \\ -0.0358979 \\ -0.0358979 \\ 0 \\ 0.0358979 \\ 0.0358979 \\ 0.0717958 \\ 0.1076937 \end{bmatrix}, \quad f2(S) = 10^{-7} * \begin{bmatrix} -0.1076937 \\ -0.0358979 \\ 0.0358979 \\ -0.0717958 \\ 0 \\ 0.1076937 \\ -0.0358979 \\ 0.0358979 \\ 0.1076937 \end{bmatrix}$$

şeklinde sonuçlar elde ederiz.



Şekil 17. $2\sin(x) + \sin(y) = 0,\ \sin(x) + 2\sin(y) = 0$ denklemleri, St çözüm tahminleri ve S çözüm için grafik

5. SONUÇLAR

- Newton ve Kiriş yöntemleri tek değişkenli lineer olmayan denklemlerin bir açık aralıktaki bütün reel sıfır yerlerini bulacak şekilde genelleştirildi.
- Dichotomous ve Dik İniş yöntemleri tek değişkenli reel değerli bir fonksiyonun bir aralıktaki bütün yerel minimumlarını bulacak şekilde genelleştirildi.
- İki bilinmeyenli Nonlineer denklem sisteminin bir dikdörtgensel bölgedeki bütün reel çözümlerini bulmak için Dik İniş yöntemi genelleştirildi.

6. ÖNERİLER

Bu çalışmada lineer olmayan tek değişkenli denklemlerin bir aralıktaki ve iki bilinmeyenli denklem sistemlerinin bir bölgedeki reel çözümlerini hesaplamak için algoritmalar geliştirildi. Benzer şekilde tek değişkenli denklemlerin ve iki bilinmeyenli denklem sistemlerinin kompleks köklerini de bulmak için algoritmalar geliştirilebilir. Ayrıca tek değişkenli denklemlerin çözüm tahminleri için verilen adaptif ağ denklem sistemleri için de genelleştirilebilir.

7. KAYNAKLAR

1. Atkinson, K. E. , An Introduction to Numerical Analysis, Second Edition, John Wiley & Sons, Inc., U.K., 1989.

2. Bayram, M., Nümerik Analiz, Birsen Yayınevi, İstanbul, 2009.

3. Fletcher, R. , Practical Methods of Optimization, Second Edition, John Wiley & Sons , 1987.

4. Mathews, J. H. and Fink, K. D., Numerical Methods Using Matlab, Third Edition, Prentice Hall, 1999.

8. EKLER

Ek 1. Sıfır Yeri Tahmin Algoritması İçin MATLAB/OCTAVE Program Kodu (fsifirtahmin.m)

```
function St=fsifirtahmin(f,a,b,dx)
% Bu program verilen bir fonksiyonun (a,b) aralığındaki
% bütün reel sıfır yerleri için birer tahmin belirler.
% f   : sıfır yerleri belirlenecek fonksiyon
% a   : aralığın sol ucu
% b   : aralığın sağ ucu
% dx  : En büyük adım uzunluğu
nn=nargin;
if nn==3
   dx=(b-a)/20;
end
f=char(f);f=vectorize(f);f=inline(f);
ezplot(f,[a,b]);hold on;
X=a;i=1;M=100;c=5;
while X(end)<b
   fx=abs(f(X(i)));
   X=[X,X(i)+dx*(1+c*fx)/(M+c*fx)];
   i=i+1;
end
if X(end)<b
   X=[X,b];
else
   X(end)=b;
end
y=abs(f(X));
yp=diff(y);
ypi=sign(yp);
test=diff(ypi);
i=find(test~=0);
St=X(i)';
if length(St)>0
   plot(St,0,'ko');
end
```

Ek 2. Vektör Tabanlı Newton Yöntemi İçin MATLAB/OCTAVE program kodu (vnewton.m)

```
function Xf=vnewton(f,X0,a,b,df,imax,eps)
% Bu program verilen bir tek değişkenli fonksiyon ve
% başlangıç değerleri için (a,b) aralığındaki kökleri
% Newton yöntemi ile vektörel formda hesaplar.
% f   : sıfır yerleri belirlenecek fonksiyon
% X0  : başlangıçdeğeri
% a   : aralığın sol ucu
% b   : aralığın sağ ucu
% df  : f'nin türevi
% eps : sonunuçlandırma kriteri için tolerans
% imax: maksimum iterasyon sayısı
nn=nargin;
if nn==6
   eps=1e-7;
elseif nn==5
   eps=1e-7;imax=100;
elseif nn==4
   eps=1e-7;imax=100;df=diff(f);
else
   eps=1e-7;imax=100;df=diff(f);a=-inf;b=inf;
end
boy=size(X0);
if boy(1)==1
   X0=X0';
end
f=char(f);df=char(df);
df=vectorize(df);df=inline(df);
f=vectorize(f);f=inline(f);
fark=1;X=[];iter=0;
while iter<imax & length(X0)~=0
   df0=find(df(X0)==0);%türevin sıfır olduğu indisleri bul
   df1=find(df(X0)~=0);%türevin sıfır olmadığı indisleri bul
   f0=find(f(X0)==0);
   X=[X;X0(intersect(f0,df0))];
   X0=X0(df1);%türevin sıfır olmadığı noktalarla devam et
   X1=X0-f(X0)./df(X0);
   fark=abs(X1-X0);
    Yn=find(fark>eps);%yakınsamayan indisleri bul
    Y=find(fark<=eps);%yakınsayan indisleri bul
   X=[X;X1(Y)];%yakınsayanları X'e at
   X0=X1(Yn);%sadece yakınsamayanlarla devam et
    iter=iter+1;
end
X=sort(real(X))';
X=X(find(X>=a));
```

Ek 2 devamı

```
if ~isempty(X)
X=sort(X)';Xf=X(1);
for i=2:length(X);
    if X(i)>b break;
    end
    if abs(X(i)-X(i-1))>eps*10
        Xf=[Xf;X(i)];
    end
end
Xf=Xf;
else
    Xf=[];
end
plot(Xf,0,'k*')
```

Ek 3. Vektör Tabanlı Kiriş Yöntemi İçin MATLAB/OCTAVE Program kodu (vkiris.m)

```
function Xf=vkiris(f,X0,a,b,imax,eps)
% Bu program verilen bir tek değişkenli fonksiyon ve
% başlangıç değerleri için (a,b) aralığındaki kökleri
% Kiriş yöntemi ile vektörel formda hesaplar.
% f   : sıfır yerleri belirlenecek fonksiyon
% X0  : başlangıçdeğeri
% a   : aralığın sol ucu
% b   : aralığın sağ ucu
% eps : sonunuçlandırma kriteri için tolerans
% imax: maksimum iterasyon sayısı
nn=nargin;
if nn==5
   eps=1e-7;
elseif nn==4
   eps=1e-7;imax=30;
else
   eps=1e-7;imax=30;a=-inf;b=inf;
end
boy=size(X0);
if boy(1)==1
   X0=X0';
end
f=char(f);f=vectorize(f);f=inline(f);
pert=0.1;
X1=X0+pert;
i=0;X=[];
while i<imax & length(X0)~=0
   df=(f(X1)-f(X0))./(X1-X0);
   i0=find(df~=0);X0=X0(i0);X1=X1(i0);df=df(i0);%turevi sıfırdan farklı başlangıç
değerleri
   X2=X1-f(X1)./df;
   fark=abs(X2-X1);
   j0=find(fark<=eps);%Yakınsayan indisler
   j1=find(fark>eps);%Yakınsamayan indisler
   X=[X;X2(j0)];%Yakınsayanları X'e at
   X0=X1(j1);X1=X2(j1);%Yakınsamayanlarla devam et
   i=i+1;
end
X=sort(X)';
X=X(find(X>=a));
if ~isempty(X)
X=sort(X)';Xf=X(1);
for i=2:length(X);
```

Ek 3 devamı

```
      if X(i)>b break;
   end
   if abs(X(i)-X(i-1))>eps*10;
       Xf=[Xf;X(i)];
   end
end
else
   Xf=[];
end
plot(Xf,0,'k*')
```

Ek 4. Yerel Minimum Tahmin Yöntemi İçin MATLAB/OCTAVE Program Kodu (fmintahmin.m)

```
function Sta=fmintahmin(f,a,b,dx)
% Bu program verilen bir fonksiyonun (a,b) aralığındaki
% her bir yerel minimum için birer tahmin aralığı belirler.
% f   : minimumları bulunulacak fonksiyon
% a   : aralığın sol ucu
% b   : aralığın sağ ucu
% dx  : En büyük adım uzunluğu
nn=nargin;
if nn==3
   dx=(b-a)/20;
end
f=char(f);f=vectorize(f);f=inline(f);
ezplot(f,[a,b]);hold on;
X=a;i=1;M=10;c=2;
while X(end)<b
   dfx=abs(df(f,X(i),dx));
   X=[X,X(i)+dx*(1+c*dfx)/(M+c*dfx)];
   i=i+1;
end
if X(end)<b
   X=[X,b];
else
   X(end)=b;
end
y=f(X);
yp=diff(y);
ypi=sign(yp);
test=diff(ypi);
i=find(test>0);
Sta=[X(i)',X(i+2)'];
if length(Sta)>0
   plot(Sta(:,1),0,'k<');
   plot(Sta(:,2),0,'k>');
end
function dfx=df(f,x,dx)
h=dx*0.01;
dfx=(f(x+h)-f(x))/h;
```

Ek 5. Vektör Tabanlı Dichotomous Yöntemi İçin MATLAB/OCTAVE Program Kodu (vdichotomous.m)

```
function S=vdichotomous(f,Sta,eps)
% Bu program verilen bir tek değişkenli fonksiyon ve başlangıç
% aralıkları için (a,b) aralığındaki yerel minimumları dichotomous
% yöntemi ile vektörel formda hesaplar.
%f   : Yerel minimumları belirlenecek fonksiyon
% Sta : başlangıç aralıkları
% eps : sonunuçlandırma kriteri için tolerans
nn=nargin;
if nn==2
   eps=1e-6;
end
f=char(f);f=vectorize(f);f=inline(f);
S=[];
while Sta(:,2)-Sta(:,1)>eps
   c=(Sta(:,1)+Sta(:,2))/2;
   j1=find(f(c-eps)<f(c+eps));
   j2=find(f(c-eps)>f(c+eps));
   j3=find(f(c-eps)==f(c+eps));
   Sta(j1,2)=c(j1);
   Sta(j2,1)=c(j2);
   Sta(j3,:)=[];
   S=[S;c(j3)];
   yi=find((Sta(:,2)-Sta(:,1))<eps);
   c=(Sta(yi,1)+Sta(yi,2))/2;
   S=[S;c];
   Sta(yi,:)=[];
end
S=sort(S);
plot(S,0,'k*');
```

Ek 6. Tek Değişkenli Fonksiyonların Minimizasyonunda Vektör Tabanlı Dik İniş Yöntemi İçin MATAB/OCTAVE Program kodu (vdikinis.m)

```
function Xf=vdikinis(f,X0,a,b,df,imax,eps)
% Bu program verilen bir tek değişkenli fonksiyon ve başlangıç
% değerleri için (a,b) aralığındaki yerel minimumları Dik iniş
% yöntemi ile vektörel formda hesaplar.
% f   : Yerel minimumları belirlenecek fonksiyon
% X0  : başlangıç değerleri
% a   : aralığın sol ucu
% b   : aralığın sağ ucu
% df  : f'nin türevi
% eps : sonunuçlandırma kriteri için tolerans
% imax: maksimum iterasyon sayısı
nn=nargin;
if nn==6
   eps=1e-6;
elseif nn==5
   eps=1e-6;imax=50;
elseif nn==4
   eps=1e-6;imax=30;df=diff(f)
else
   eps=1e-6;imax=50;df=diff(f);a=-inf;b=inf;
end
boy=size(X0);
if boy(1)==1
   X0=X0';
end
f=char(f);df=char(df);
df=vectorize(df);df=inline(df);
f=vectorize(f);f=inline(f);
ezplot(f,[a,b]);hold on
X=[];iter=0;alfa=1;
A=alfa.*ones(length(X0),1);
while iter<imax & length(X0)~=0
   df0=find(abs(df(X0))<eps);
   df1=find(abs(df(X0))>=eps);
   X=[X;X0(df0)];
   X0=X0(df1);A=A(df1);
   X1=X0-A.*df(X0)./abs(df(X0));
   ay=find(f(X1)>f(X0));
   while length(ay)>0
      A(ay)=A(ay)./2;
      X1=X0-A.*df(X0)./abs(df(X0));
      ay=find(f(X1)>f(X0));
   end
   Ui=find(A>=eps/100);
   X0=X1(Ui);
```

Ek 6 devamı

```
    iter=iter+1;
end
X=sort(X)';
X=X(find(X>=a));
if ~isempty(X)
X=sort(X)';Xf=X(1);
for i=2:length(X);
    if X(i)>b break;
    end
    if abs(X(i)-X(i-1))>eps*10
        Xf=[Xf;X(i)];
    end
end
else
    Xf=[];
end
plot(Xf,0,'k*');
```

Ek 7. Denklem Sistemi Çözüm Tahmini İçin MATLAB/OCTAVE Program Kodu (fsifirtahmindenksis.m)

```
function St=fsifirtahmindenksis(f1,f2,X,Y,dx,dy)
% Bu program iki bilinmeyenli denklem sisteminin
% XxY bölgesindeki bütün ayrık çözümleri için tahminler belirler.
% f1,f2: Denklem sistemi için fonksiyonlar
% X    : Bölge için  X=(X1,X2) aralığı
% Y    : Bölge için  Y=(Y1,Y2) aralığı
% dx   : x-ekseni için artım miktarı
% dy   : y-ekseni için artım miktarı
nn=nargin;
if nn==4
   dx=(X(2)-X(1))/100;
   dy=(Y(2)-Y(1))/100;
elseif nn==5
   dy=dx;
end
f11=vectorize(f1);f22=vectorize(f2);
f11=inline(f11);f22=inline(f22);
ezplot(f1,[X(1),X(2)],[Y(1),Y(2)]);hold on
ezplot(f2,[X(1),X(2)],[Y(1),Y(2)]);
x=X(1):dx:X(2);
y=Y(1):dy:Y(2);
[x1,y1]=meshgrid(x,y);
Z=F(x1,y1,f11,f22);
dFx=diff(Z,1,2);
dFxi=sign(dFx);
dFxid=diff(dFxi,1,2);
dFy=diff(Z,1,1);
dFyi=sign(dFy);
dFyid=diff(dFyi,1,1);
test=dFxid(1:end-2,:).*dFyid(:,1:end-2);
[i,j]=find(test~=0);
St=[x(j)',y(i)'];
if length(St)>0
plot(St(:,1),St(:,2),'ko');
end
function f=F(x,y,f11,f22)
f=(f11(x,y)).^2+(f22(x,y)).^2;
```

Ek 8. Denklem Sisteminin Çözümü İçin Vektör Tabanlı Dik İniş Yöntemi MATLAB/OCTAVE Program Kodu (vdikinisdenk.m)

```
function Si=vdikinisdenk(f11,f22,St,df,eps,imax)
% Bu program iki bilinmeyenli denklem sisteminin çözümünü
% Dik İniş yöntemi ile vektörel formda hesaplar.
% f11,f22: Denklemler
% St     : Başlangıç değerleri
% df     : f11 ve f22'nin gradyanı
% eps    : sonunuçlandırma kriteri için tolerans
% imax   : maksimum iterasyon sayısı
nn=nargin;
if nn==3
   sy=symvar(char(f11));x=sym(char(sy(1)));y=sym(char(sy(2)));
   df1=jacobian(f11,[x,y]);
   df2=jacobian(f22,[x,y]);
   df={char(df1(1)),char(df1(2));char(df2(1)),char(df2(2))};
   eps=1e-9;imax=100;
elseif nn==4
   eps=1e-8;imax=100;
elseif nn==5
   imax=100;
end
sy=symvar(char(f11));x=char(sy(1));y=char(sy(2));
df11=inline(vectorize(df(1,1)),x,y);
df12=inline(vectorize(df(1,2)),x,y);
df21=inline(vectorize(df(2,1)),x,y);
df22=inline(vectorize(df(2,2)),x,y);
f11=vectorize(f11);f22=vectorize(f22);
f11=inline(f11);f22=inline(f22);
Sn=size(St);Si=[];k=1;
A=ones(Sn(1),2);
Fd=F(St(:,1),St(:,2),f11,f22);
while length(St(:,1))>0
   NGd=2*sqrt((f11(St(:,1),St(:,2)).*df11(St(:,1),St(:,2))+...
      f22(St(:,1),St(:,2)).*df21(St(:,1),St(:,2))).^2+...
      (f11(St(:,1),St(:,2)).*df12(St(:,1),St(:,2))+...
      f22(St(:,1),St(:,2)).*df22(St(:,1),St(:,2))).^2);
   Fd=F(St(:,1),St(:,2),f11,f22);
   G0=find(NGd<eps^2);
   F0=find(abs(Fd)<eps^2);gf0=union(G0,F0);
   Si=[Si;St(F0,:)];c=setdiff(1:size(St(:,1)),gf0);
   St=St(c,:);
   A=A(c,:);NGd=[NGd,NGd];Fd=Fd(c,:);NGd=NGd(c,:);
   S1=St-A.*[2*(f11(St(:,1),St(:,2)).*df11(St(:,1),St(:,2))+...
      f22(St(:,1),St(:,2)).*df21(St(:,1),St(:,2))),2*...
      (f11(St(:,1),St(:,2)).*df12(St(:,1),St(:,2))+...
      f22(St(:,1),St(:,2)).*df22(St(:,1),St(:,2)))]./NGd;
```

Ek 8 devamı

```
   Fd1=F(S1(:,1),S1(:,2),f11,f22);
ay=find(Fd1>=Fd);k1=0;
   while length(ay)>0 & k1<imax
      A(ay,:)=A(ay,:)./2;
      S1(ay,:)=St(ay,:)-A(ay,:).*[2*(f11(St(ay,1),St(ay,2)).*...
         df11(St(ay,1),St(ay,2))+f22(St(ay,1),St(ay,2)).*...
         df21(St(ay,1),St(ay,2))),2*(f11(St(ay,1),St(ay,2)).*...
         df12(St(ay,1),St(ay,2))+f22(St(ay,1),St(ay,2)).*...
         df22(St(ay,1),St(ay,2)))]./NGd(ay,:);
      Fd1(ay,:)=F(S1(ay,1),S1(ay,2),f11,f22);
      ay=find(Fd1>=Fd);
      k1=k1+1;
   end
   Ui=find(A(:,1)>(eps^2)/2);
   St=S1(Ui,:);
   if k==imax break
   end
   k=k+1;
end
eps=eps*10;
Si=unique(round(Si*(1/eps))*eps,'rows');
plot(Si(:,1),Si(:,2),'k*','MarkerSize',5);
function f=F(x,y,f11,f22)
f=f11(x,y).^2+f22(x,y).^2;
```

ÖZGEÇMİŞ

1983 yılında Rize'nin Pazar ilçesinde doğdu. İlk, orta ve lise öğrenimini Pazar da tamamladı. 2003 yılında girdiği Karadeniz Teknik Üniversitesi Fen Edebiyat Fakültesi Matematik Bölümünü 2007 yılında bitirdi. 2009 yılında KTÜ Fen Bilimleri Enstitüsü Matematik Anabilim dalında yüksek lisans öğrenimine başlayan Murat MEMOĞLU orta derecede İngilizce bilmektedir.