2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

1.DERS:5 Mart 2021

Arş. Gör. Alev KAYA

DERS HAKKINDA GENEL BİLGİLER

- Sevgili öğrenciler,
- **Ders sorumlusu**: **Dr. Öğr. Üyesi Bihter Daş**
- **LAB: Arş. Gör. Alev Kaya**
- Her haftaya ilişkin ders videoları UE sistemine haftalık olarak yüklenecektir.
- Ders ve Lab uygulamaları MATLAB platformunda gerçekleştirilecektir.
- Derse ilişkin sorularınızı UE sistemi üzerinden gönderiniz.

DEĞERLENDİRME

- **VİZE**=(Ödev*0.4+Vize sınavı*0.6)
- **FİNAL**= Çoktan seçmeli ve boşluk doldurmalı sorulardan oluşacaktır.
- **BÜTÜNLEME**=Çoktan seçmeli ve boşluk doldurmalı sorulardan oluşacaktır

DERSİN UYGULAMASI:
1.Hafta: MATLAB Hakkında Genel Bilgiler

Kapsam

- 1-MATLAB nedir?
- 2-Ne işe yarar?
- 3-Mühendislikte nerede, nelerde kullanabiliriz.
- 4-MATLAB nasıl kurulur?

1-MATLAB(**MAT**rix **LAB**oratory) **NEDİR?**

- **MATLAB**; 1985 yılında **C.B.Moler** (Matematikçi ve bilgisayar programcısı) tarafından geliştirilen bir paket programı,
- **MAT**rix L**AB**oratory sözcüklerinin ilk 3 harflerinin birleşimiyle isimlendirilmiştir,
- Matematik ve özellikle **matris tabanlı** matematiksel işlemler için geliştirilmiş.
- **İlk sürüm FORTRAN** dilinde,
- Daha **sonraki yıllarda C diliyle** hazırlanmış,
- **İki sürümü mevcut: a ve b**
- **a: alfa, son sürüm: asıl sürüm**
- **b: beta, ilk çıkan yani tanıtım sürümüdür.**
- 2013 sürümünden bu yana bu görüntüyle çalışıyor.
- Eğer bir kullanıcı programlama dilleri hakkında yeterli bilgisi yoksa MATLAB la başlayabilir!!!(Yazılımcıları kapsamıyor…)

1-MATLAB(MATrix LABoratory) NEDİR?

- Bir çok klasik algoritma, bir komut ile kullanıcıya sunulması(önemli üstünlük),
- Böylece çok sayıda satıra sahip programlar kısalır,
- Algoritmanın sona erme süresi ve hem de bellek ihtiyacı azalır,
- Geleneksel prog. dillerinin aksine programı derleyip(compile) çalıştırabilir,
- Bir dosya(exe) haline getirmeden, yorumlayarak(interpreter) çalıştırır,
- Böylece programın hatalardan arındırılması sürecinde ciddi zaman tasarrufu,
- Program satırları MATLAB' da iki farklı ortamda yazılabilir.
- Birincisi; Command Window: Komut Penceresi ortamında
- İkincisi; MATLAB edit(editör) ortamı olarak adlandırılan 'M file' dosya editörü.

2- MATLAB NE İŞE YARAR?

- Mühendislik alanındaki hesaplamalarda(alg. geliştirmede),
- Sayısal hesaplamalarda[İntegral hesabı, türev alma, matris işlemleri],
- Veri ve fonksiyon çözümlemelerinde,
- Denklem takımlarının çözümü,
- Doğrusal ve doğrusal olmayan diferansiyel denklemlerin çözümü,
- İstatistiksel hesaplamalar ve çözümler,
- Grafik çizimleri(2d,3d),
- Bilgisayar destekli denetim sistemleri,
- Kullanıcı arayüz oluşturmada,
- C,C++,Java,Python,… gibi diğer dillerde yazılmış programlarla birlikte çalışma imkanı sağlar.

3-MÜHENDİSLİKTE NEREDE, NERELERDE KULLANABİLİRİZ

- Matematik,
- Görüntü ve Sayısal İşaret İşleme,Sinyal İşleme,
- Finans ve İstatistik, Optimizasyon
- Yapay Zeka Uygulamalarında,**(Hazır toolbox: araç kutuları mevcut)**
- Yapay Sinir Ağlarında, Bulanık Kontrol
- Genetik Algoritma,
- Veri Analizinde, Grafiklerde,
- Mühendislik,
- Kontrol Sistemleri
- Güç sistemleri ve Filtre dizaynı
- Web sunucusu, Veri tabanı

4-MATLAB NASIL KURULUR (2 seçenekli)

- **1.SEÇENEK: FU LİSANSLI**
- Google ' a: http://www.firat.edu.tr yazalım
- Fırat Üniversitesi Anasayfasında '**BİLGİ SİSTEMLERİ** ' sekmesine tıklayalım,
- Bilgi Sistemleri sekmesinde '**DOSYA İNDİRME SİSTEMİ**' sekmesine tıklayalım,
- Gelen pencerede Fırat Üniversitesinin lisanslı programları mevcut.
- '**Matlab**' sekmesine tıklayıp işletim sisteminize göre(2014 b ve 2015 a sürümleri)

4-MATLAB NASIL KURULUR
1.SEÇENEK: FU LİSANSLI
Matlab R2015a Windows 32-bit

- **Önemli Not:** Kampüs dışından kullanımda hata alanlar aşağıdaki adımları uygulayınız:
- 1. "C:\Program Files\MATLAB\R2015a\licenses" klasörüne gidiniz
- 2. "network.lic" dosyasını masaüstüne kopyalayınız
- 3. Masaüstündeki"network.lic" dosyasını açınız ve "yazilim_lisans" yazan yeri silip yerine "193.255.124.125" yazınız ve dosyayı kaydediniz.
- 4. Masaüstündeki "network.lic" dosyasını "C:\Program Files\MATLAB\R2015a\licenses" klasörüne kopyalayınız.
- **NOT: LÜTFEN İŞLEM ADIMLARINI OKUYARAK GİDİNİZ. YOKSA İNDİRMEDE SORUN YAŞARSINIZ.**

4-MATLAB NASIL KURULUR
2.SEÇENEK: ONLİNE KURULUM

- **1.SEÇENEK: ONLİNE 1 AYLIK ÜCRETSİZ DENEME**
- Google ‘ a: ‘ **Matworks**’ yazalım,
- https://www.mathworks.com adresine tıklayalım.

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

2.DERS:12 Mart 2021

Arş. Gör. Alev KAYA

GENEL MATLAB ORTAMI

- **ADRES ÇUBUĞU:** MATLAB da çalıştığınız dizini gösterir. Yani çalışmalarımızı tuttuğumuz ve organize edebildiğimiz yer. Herhangi bir çalışma dizininde bir klasör oluşturup adres çubuğuna koplayıp «ENTER» larsanız, size artık o çalışma dizini altında ayrı bir yer sunar.

- **CURRENT WİNDOW:** Harddiskte kayıtlı dosyaları gösterir. Yani Çalışma Dizindeki bütün dosyaları, komutları, verileri, fonksiyonları, …bu pencereden görebilirsiniz.

- **WORKSPACE:** MATLAB içerisinde oluşturduğumuz değişkenleri ve aldıkları değerleri hemen inceleyip bu pencerede görebiliriz.

- **COMMAND WİNDOW:** Doğrudan komutlarımızı yazdığımız penceredir. MATLAB' ın yorumlayıcısında herhangi bir şekilde Compile (Derleme) yapmanıza gerek yok. Doğrudan MATLAB' a komut vererek sonuç alabiliyorsunuz.

GENEL MATLAB ORTAMI

- **HELP:** MATLAB içinde tanımlı olan komut, fonksiyon, hazır toolbox, simulink,… hakkında açıklayıcı bilgiler ve örnekler sunar. Hiçbir bilginiz olmasa dahi MATLAB kendi içerisinde güçlü bir dökümantasyon hazırlayıp bunları örneklerle pekiştirerek kullanıcıya sunmaktadır. Help ifadesi ekrana yazılıp komut da yazıldığında enter tuşuna basarak hakkında bilgi alınabilir;

Örnek: help plus, help general, help ops, help lang, help elmat,help elfun,

help specfun, help help, helpwin, helpdesk,

Yada **help(konu-komut)** hakında bilgi alınabilir.

- **Ctrl ve c:** MATLAB programı kesilir ama programdan çıkılmaz. Yada programdaki tüm değişkenleri silmek için kullanılır.
- **quit:** MATLAB programından çıkılır.
- **exit:** MATLAB programından çıkılır.

MATLAB DİLİ KULLANILARAK YAZILMIŞ BİR PROGRAMIN İCRASI NASIL OLUYOR?

- Tek satırlı komutlar yazılıp icra edildiğinde ekran üzerinde bu komutun sonucu hemen görülebilir. İkinci bir komut yazıldığında ilk komut hafızada tutulur. MATLAB yüksek seviyeli bir programlama dilidir.
- Eğer girilen komut hatalı ise hata mesajı derhal ekrana yazılır.
- Bu dilde diğer yüksek seviyeli dillerin aksine derleme(execute) ve bağlama-yükleme(link-load) işlemlerine ihtiyaç duyulmaz.
- **1.ADIM:** Yazılan programda komut ile karşılaşıldığında bu komut uygulanır.
- **2.ADIM:** Eğer komut kurallara uygun bir şekilde yazılmamış ise MATLAB programı bu komutun yazılı olduğu satırda uygulamayı durdurur ve ekrana hata mesajını (syntax error)yazar.
- **3.ADIM:** Eğer yazılımdaki bu hata giderilir ise kullanıcı tarafından program en baştan tekrar çalışmaya başlatılmalıdır.
- **4.ADIM:** Yazılan programın çalıştırılması sonunda elde edilen sonuçlar beklenen sonucun çok altında veya üstünde ise bu durumda programda mantık hatası aranmalı ve giderildikten sonra program tekrar çalıştırılmalıdır.
- **NOT: Yukarıda belirtilen adımlar yüksek seviyeli dillerde karşılaşılan derleme / bağlama/ yükleme/ icra adımlarına karşılıktır. MATLAB bu adımları ayrı ayrı komut kullanarak gerçekleştirmediği için çok verimli ve kullanımı kolay bir programlama dilidir.**

MATLABDA GENEL KOMUTLAR

- **% (YÜZDE İŞARETİ):** Bulunduğu satırı açıklama(yorum) için kullanılır. Genelde kodlar yazılmadan önce satır başlarında gerekli açıklamalar yapılırsa konu ile alakalı, daha objektif şekilde ilerlenebilir.ve kodunuzun ne için yazıldığı hem sizin tarafınızdan hem de bir başkası tarafından geriye dönüp bakıldığında daha açıklayıcı bilgilere sahip olunur.

Örnek: % Bu slaytta MATLAB paket programı ile ilgili bilgiler verilecektir.

- **;(NOKTALI VİRGÜL):** Bu işaretin bulunduğu satırda elde edilen sonuc, programın icrası sırasında ekranda görünmesini engeller.

Örnek: x=5;

- **…(ÜÇ NOKTA):** Eğer bir satırın sonunda üç nokta varsa bu satırın alttaki satırda devam ettiği anlaşılır.

Örnek: x=andgbskmkjjvjnnskkfnvlsjknbvbnmdnmmmkskbd…

bhabdjnckjkmllkldvbhbbbbjndkmmllşlslknvhbs1454515454

- **,(VİRGÜL):**Birden çok değişken aralarına virgül konularak aynı satır içinde yan yana yazılabilir.
- **Örnek:** x=5, y=8, z=12

HESAPLAMA İŞLEMLERİNDE KULLANILAN KAVRAMLAR

KOMUT (COMMAND): Kullanıcının bilgisayara uygulaması için verdiği emirlerdir. Bunlar kullanılan dilin alt yapısında tanımlı olduğu için program tarafından icra edilir.

- **DEĞİŞKEN (VARİABLE):** Bir büyüklüğe verilen isimdir. Değişkene programın akışı sırasında sayısal yada alfa sayısal bir değer karşılık gelir. Eşitlik işaretinin sol tarafındaki harf yada kelime değişken ismi olarak atanır. Bu harf(yada kelime), eşitlik işaretinin sağ tarafındaki değere karşılık gelir.

- **VARSAYILAN (DEFAULT):** Programın icrası sırasında aradığı değeri bulamadığında sistemin içinde mevcut seçeneklerden genelde kabul edileni seçmesi.

- **KONUM DEĞİŞİMİ (TOGGLE):** İki değer alabilen bir değişkenin içinde bulunduğu durumdan çıkıp diğer konuma geçmesi.

- **ARGÜMAN(ARGUMET):** Bir komutun uygulandığı değişken.
- **İCRA (EXECUTE):** Yazılı komutların uygulanması.
- **EKRAN (DİSPLAY):** Yazılımla ilgili bilgilerin ekrandan izlenmesi.
- **YAZMA (PRİNT):** Yazıcı yardımı ile istenilen bilgilerin yazdırılması.

MATLABDA ÖZEL TANIMLAR

- **>>:** MATLAB programının çalışmaya başladığını gösteren işarettir. Bu işaretin sağ tarafına MATLAB diliyle uyumlu komutlar yazıldığında program bunları icra etmeye (execute) başlar. MATLAB dilinde yazılan dosyaların uzantıları m olmalıdır.

Örnek: 'ılkders.m'

- **ans:** Değişken ismi verilmediğinde hesaplama sonunda elde edilen cevabı gösterir.

Örnek: 3+2 (enter) ans=5

- **pi**= π=3,1415926 değerinin kısa yazılışı
- **eps**: iki sayı arasındaki farkın en küçük değeri

Örnek: eps (enter) ans = 2.2204e-16

- **İnf(yada Inf):** Sonsuz, sıfıra bölme işleminde ortaya çıkar(sayı/0).
- **Nan(yada NAN-Not a Number):** Rakam değil, ÖRNEĞİN; 0/0 da olduğu gibi tanımlanmamış deyimlerde ortaya çıkar.
- **linspace:** Vektör uzunluğunu gösterir. Aşağıdaki örnekte 1 den başlayarak 2 ye kadar 20 aralığa bölerek sayılar üretir.

Örnek: t= linspace(1,2,20)

MATLABDA ÖZEL TANIMLAR

- **date:** Programın yazıldığı gün,ay,yıl.
- **flops:** Yüzer nokta işlemlerinin sayısını sayar.
- **who:** Bellekteki değişkenleri listeler.
- **whos:** Bellekteki değişkenlerin isim, büyüklüklerini ve kaç byte(bayt) oldukalrını listeler.
- **clear:** Bellekte saklanan değerleri listeler.
- **clc:** Komut penceresini siler, değişkenlere etki etmez.
- **clf:** Mevcut olan şekil veya grafik penceresini siler.
- **more**: Komut penceresi için denetimli sayfa çıkışı.
- **plot:** Çizim yapmada kullanılan komuttur.

İYİ BİR PROGRAMCININ UYMASI GEREKLİ KURALLAR

- Yazılan program satırlarında çok açıklayıcı (yorum) ifadeler kullanılmak,
- Programda kullanılan değişkenlerin ne anlama geldiğini açıklamak,
- Program içindeki farklı bölümleri farklı renkler kullanarak birbirinden ayırmak,
- Program içindeki döngüleri birbirlerinden ayırmak için döngünün başlangıç yerini biraz içeriye almak(bu programın lojik kontrolü için yararlı)
- …

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

3.DERS

Arş. Gör. Alev KAYA

3.HAFTA
Lineer Olmayan Denklem Sistemlerimin Çözümü

- Kapalı Yöntemler

A- Grafik Yöntemi

B- Bisection Yöntemi

LAB: Grafik yöntemi ve Bisection yöntemi örnek Matlab programı

DOĞRUSAL OLMAYAN (NONLINEAR) DENKLEM SİSTEMLERİ

Kapalı Yöntemler

- İki veya daha yüksek dereceli polinomlar veya trigonometrik, üstel ve logaritmik gibi lineer olmayan terimler içeren denklemler **lineer olmayan denklemlerdir.**

Örnek: $x\ ^3 + 2x\ ^2 - 5\ sinx = 0\ veya\ x - tanx = e\ ^{-x}$

denklemleri tek bilinmeyenli lineer olmayan denklemlerdir

- Genelde lineer olmayan denklemler $f\ (x\ ) = 0$ **kapalı formunda** yazılırlar.
- Karşılaşılan denklemlerin **çoğu tek değişkenli** olmakla beraber **çok değişkenli** $f(x1,x2,x3,\ldots) = 0$ denklemlerin çözümü de söz konusu olabilir.
- Kök bulma işlemi, verilen $f(x)$ denkleminde $f\ (xk\ ) = 0$ değerini sağlayan xk değerlerinin bulunması işlemidir.
- Tek değişkenli bir fonksiyon için bu değerler aynı zamanda **eğrinin x eksenini kestiği noktalardır.**
- Kök bulma işlemlerinde öncelikle kökün hangi aralıkta olduğu belirlenir.
- a ve b gibi iki farklı sayı ile belirlenen aralıkta $(a \le xk \le b)$ tanımlanmış $f(x)$ fonksiyonu bu aralıkta sürekli ise $f(a) \times f(b) < 0$ ise, öyle bir xk değeri vardır ki, $f\ (xk\ ) = 0$ eşitliğini sağlar.

DOĞRUSAL OLMAYAN (NONLINEAR) DENKLEM SİSTEMLERİ

A-Grafik Yöntemi

- Kök bulma işlemi denklemi sağlayan bağımsız değerlerin bulunması işlemidir denebilir.
- Sayısal çözümlemeler geliştirilmeden önce denklemlerin köklerinin bulunmasına yönelik çeşitli çözümler geliştirilmiştir.
- Mesela 2. Dereceden denklemlerin çözümü için çeşitli formüller kullanılırdı.
- Ancak birçok denklem bu şekilde basitçe çözülememekteydi.
- Bazı denklemler için analitik çözümler geliştirilememekte ve yaklaşık çözümler üretilmekteydi.
- Yaklaşık çözüm elde etmenin en pratik ve en ilkel yolu grafik yöntemidir.
- **Grafik yönteminde** fonksiyona ait bazı değerler elde edilerek grafiği çizilir.
- Çizilen grafik yardımı ile grafiğin x eksenini kestiği kök noktası tahmin edilir.

TEK DEĞİŞKENLİ DENKLEMLERİN ÇÖZÜMÜ

- Tek değişkenli $f(x) = 0$ denkleminin çözmek için değişik yöntemler kullanılmaktadır.
- Bunlar iteratif yöntemler olup kökler için tahmini değerlerin alınmasın gerektirir.
- Bu yöntemlerin bir kısmı, nonlineer denklemin yerine lineer bir denklem kabul edilip çözüme ulaşma esasına dayanır.

- Biz de yaygın olarak kullanılan;
- 1. Yarıya Bölme (Bisection)
- 2. Lineer Interpolasyon (Regula-Falsi)
- 3. Basit İterasyon
- 4. Newton-Raphson
- 5. Kiriş (Secant)

yöntemlerini inceleyeceğiz.

Örneğin: $f(x) = xe\text{^}x - 2$ ifadesini [0,1] aralığında 0.25 aralıklar ile inceleyelim:

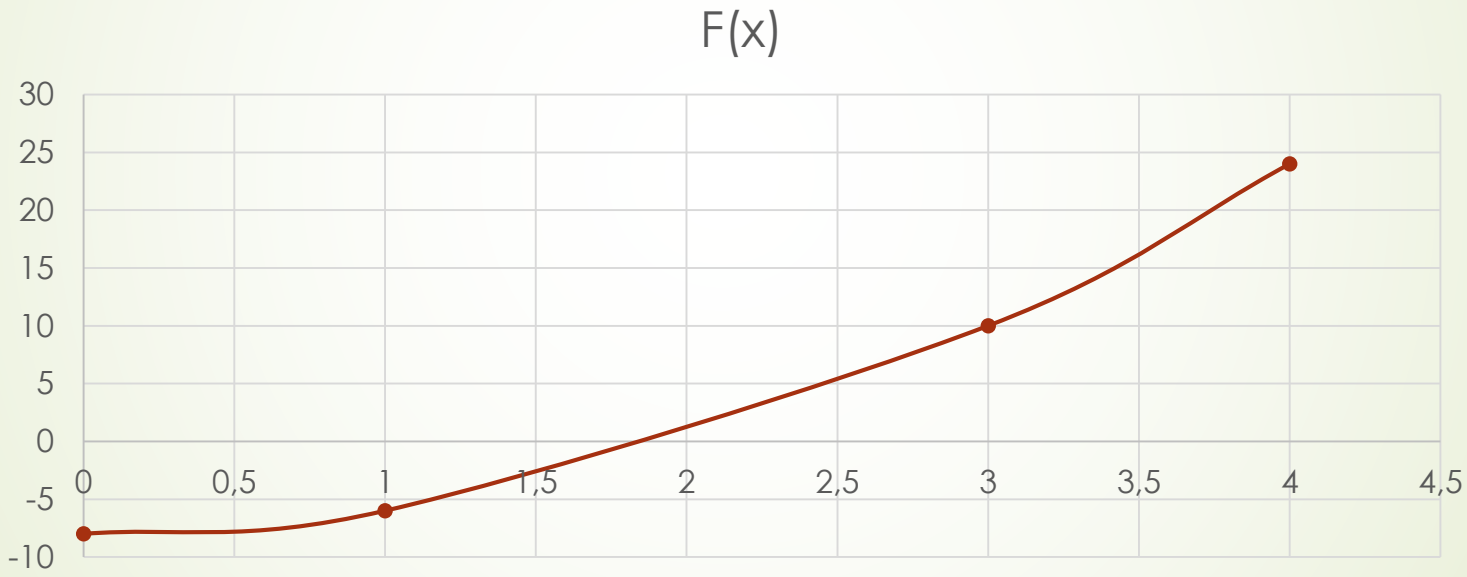
x	F(x)
0,0	-2
0,25	--1,6788993
0,5	-1,175639
0,75	-0,412250
1,0	0,718281

f (0,75) × f (1,0) < 0 olduğundan aranan kök *0.75,1.0+ aralığındadır.

f(0,85) = −0,011300 olduğundan aranan kök *0.85,1.0+ aralığındadır.

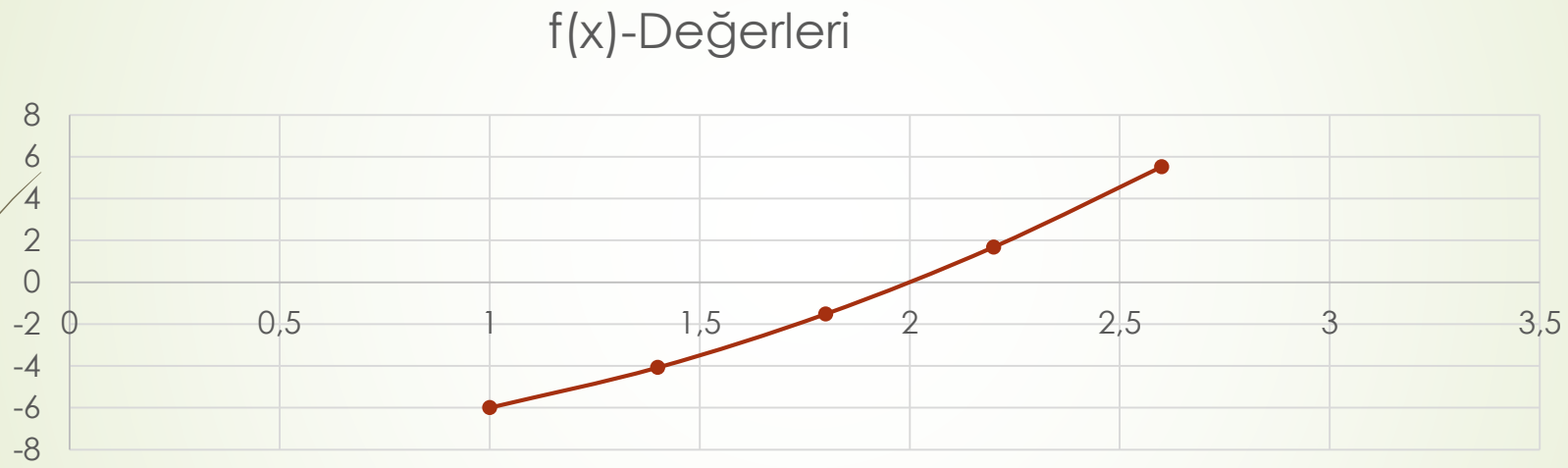
Örnek: $f\ x = 2x\ ^2 - 8$ denkleminin kökünü grafik yöntemi ile bulalım.

x	0	1	3	4
F(x)	-8	-6	10	24



İlk aşamada kökün [1,3] aralığında olduğu belirlenmiştir.
Şimdi aralığı biraz daha daraltalım.

X-1.52	1.0	1.4	1.8	2.2	2.6	3.0
F(x)	-6	-4.08	-1.52	1.68	5.52	10



Buradan da aranan kökün $x = 2$ olduğu kolayca bulunabilmektedir.
Yalnız her problemde kök bu kadar kolay bulunamayacağı açıktır.

DOĞRUSAL OLMAYAN (NONLINEAR) DENKLEM SİSTEMLERİ

B-.YARIYA BÖLME (BİSECTİON) YÖNTEMİ

- $f\ x = 0$ şeklinde bir denklem verilsin.
- $f\ (x)$ fonksiyonu [a,b] aralığında sürekli ve $f(a) \times f(b) < 0$ ise $f\ x$ fonksiyonunun (a,b) aralığında bir yada birden fazla kökü vardır.
- Bu yöntem birden fazla kök için geçerli olsa da biz $f\ (x)$ 'in (a,b) aralığında sadece bir kökünün olduğunu varsayacağız.
- $f(a) \times f(b) < 0$ olduğundan (a,b) aralığında kök vardır.
- **1. iterasyon:** $x1 = (a+b)/\ 2$;
- **2. iterasyon:** **IF** $f(a) \times f(x1) < 0$ ise

  $x2 =(\ a+x1)/\ 2$;

  ELSE

  $x2 = (b+x1)\ /2$;

- **3.iterasyon:** **IF** $f(a) \times f(x2) < 0$ ise

  $x3 = (a+x2)/\ 2$;

  ELSE

  $x3 = (x1+x2)\ /2$;

İterasyonlar istenen hata aralığına ulaşıncaya kadar devam eder.

Örnek: $f(x) = x\,\text{^}4 - 9x\,\text{^}3 - 2x\,\text{^}2 + 120x - 130$ **eşitliğinin (1,2) aralığında bir köke sahip olduğu bilinmektedir. Bu kökü** $\varepsilon y \leq 0{,}0132$ **yaklaşım hatası ile bulunuz.**

- **Çözüm:** $a = 1.0\ ve\ b = 2.0$

$a = 1.0\ iken\ f(1.0) = -20\ ve\ b = 2.0\ iken$
$f(2.0) = 46$

1.Adım : $f(a) \times f(b) = (-20) \times 46 < 0$
$olduğundan\ bu\ aralıkta\ kök\ vardır.$

$x1 = (a + b)/\ 2 = (1 + 2)/\ 2 = 1.5\ ve$

$f(1.5) = 20.2$

$b = x1 = 1.5$ olur.

2.Adım : $f(a) \times f(b) < 0\ olduğundan$

$x2 = (a + b)/\ 2 = (1 + 1.5)/\ 2 = 1.25\ ve$

$f\ (1.25) = 1.8\ olur.$

Bu durumda $b = x2 = 1.25$ olur.

- **3.Adım :** $f(a) \times f(b) < 0\ olduğundan$

$x3 = (a + b)/\ 2 = (1 + 1.25)/\ 2 = 1.125\ ve$

$f\ (1.125\ ) = -8.7$

$a = x3 = 1.125$ olur.

4.Adım : $f(a) \times f(b) < 0\ olduğundan$

$x4 = (a + b)/\ 2 = (1.125 + 1.25)/\ 2 = 1.1875\ ve$

$f\ (1.1875\ ) = -3.4028$

$a = x4 = 1.1875$ olur.

DEVAMI

5.Adım : $f(a) \times f(b) < 0$ *olduğundan*

$x5 = (a + b)/\ 2 = (1.1875 + 1.25)/\ 2 = 1.21875$ *ve*

$f\ (1.21875\ ) = -0.80688$

$a = x5 = 1.21875$ olur.

6.Adım : $f(a) \times f(b) < 0$ *olduğundan*

$x6 = (a + b\ )/2 = (1.21875+1.25)/\ 2 = 1.234375$

ve

$f(x6) = 0{,}472092$

$b = x6 = 1.234375$ olur.

$\varepsilon y \leq \left| (1.234375 - 1.21875)/\ 1.234375 \right| = 0{,}01265$

- ALGORİTMA
- 1. $IF\ f(a) \times f(b) < 0$
- 3. $REPEAT$
- 4. $xk = (a+b)/2\ ;$
- 5. $IF\ f(a) \times f(xk\ ) < 0$
- 6. $b = xk$
- 7. $ELSE$
- 8. $a = xk$
- 9. $Hatayı\ Hesapla\ (\varepsilon)$
- 9. $UNTIL\ (\varepsilon \leq Hata\ Toleransı)$
- 10. $ELSE$
- 11. " $(a,\ b)\ aralığında\ kök\ yoktur.$ "

Teorem: $f \in C\ [a, b]\ ve\ f\ (a\ )*\ f\ (b) < 0$ olsun.

- Bu duruma aralık yarılama yöntemi ile f'in kökü xr 'ye yaklaşan bir $\{xn\ \}n$=1 den ∞ dizisi oluşturur.

- Oluşabilecek maksimum hata;

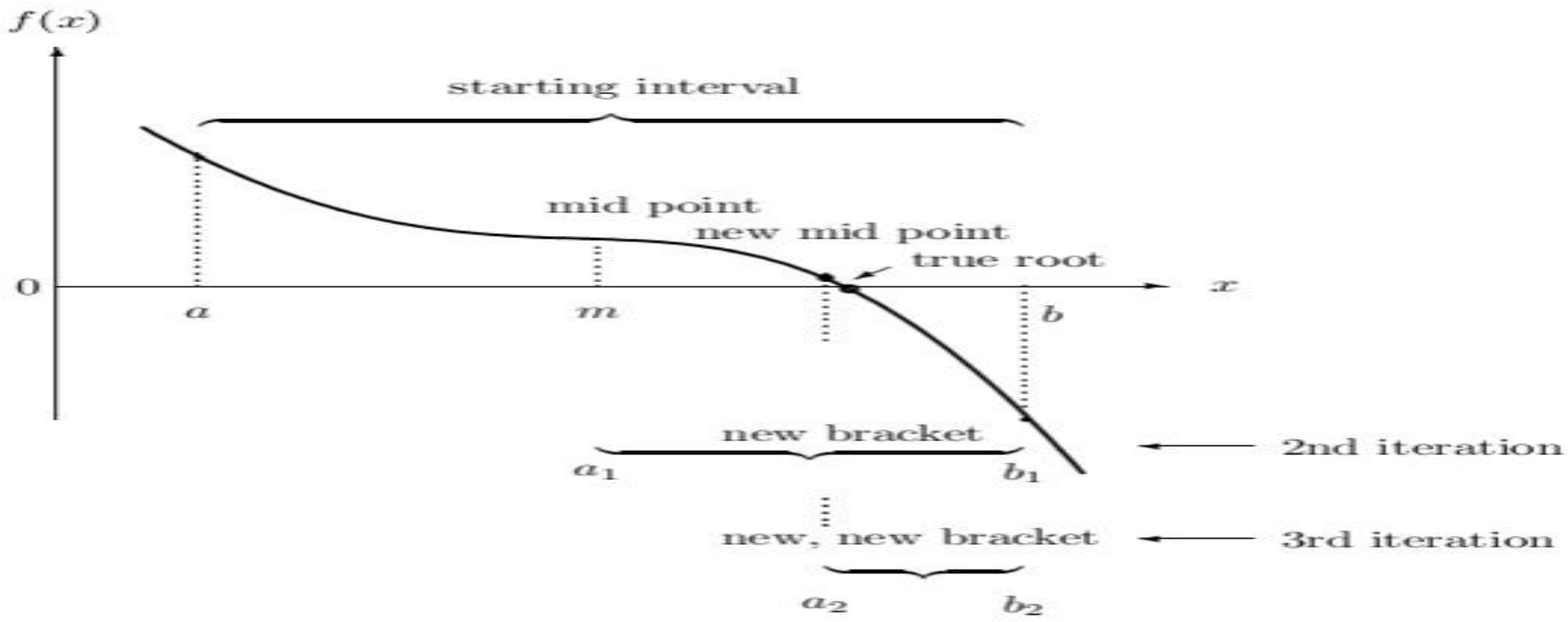
$$\varepsilon \leq (b - a)/\ (2\hat{}\ n)$$

- Oluşabilecek maksimum hata ise;

$\varepsilon max = (b - a)/\ (2\ \hat{}n)$ şeklinde hesaplanır.

İKİYE BÖLME YÖNTEMİ (BISECTION METHOD)

Eğer f(x)=0 denkleminin (a,b) aralığında kökü olması için f(a).f(b)<0 koşulu sağlanması gerekmektedir.



Yöntem için adımlar aşağıdaki gibidir:

Algoritma

1.Adım: i=1 olarak belirle

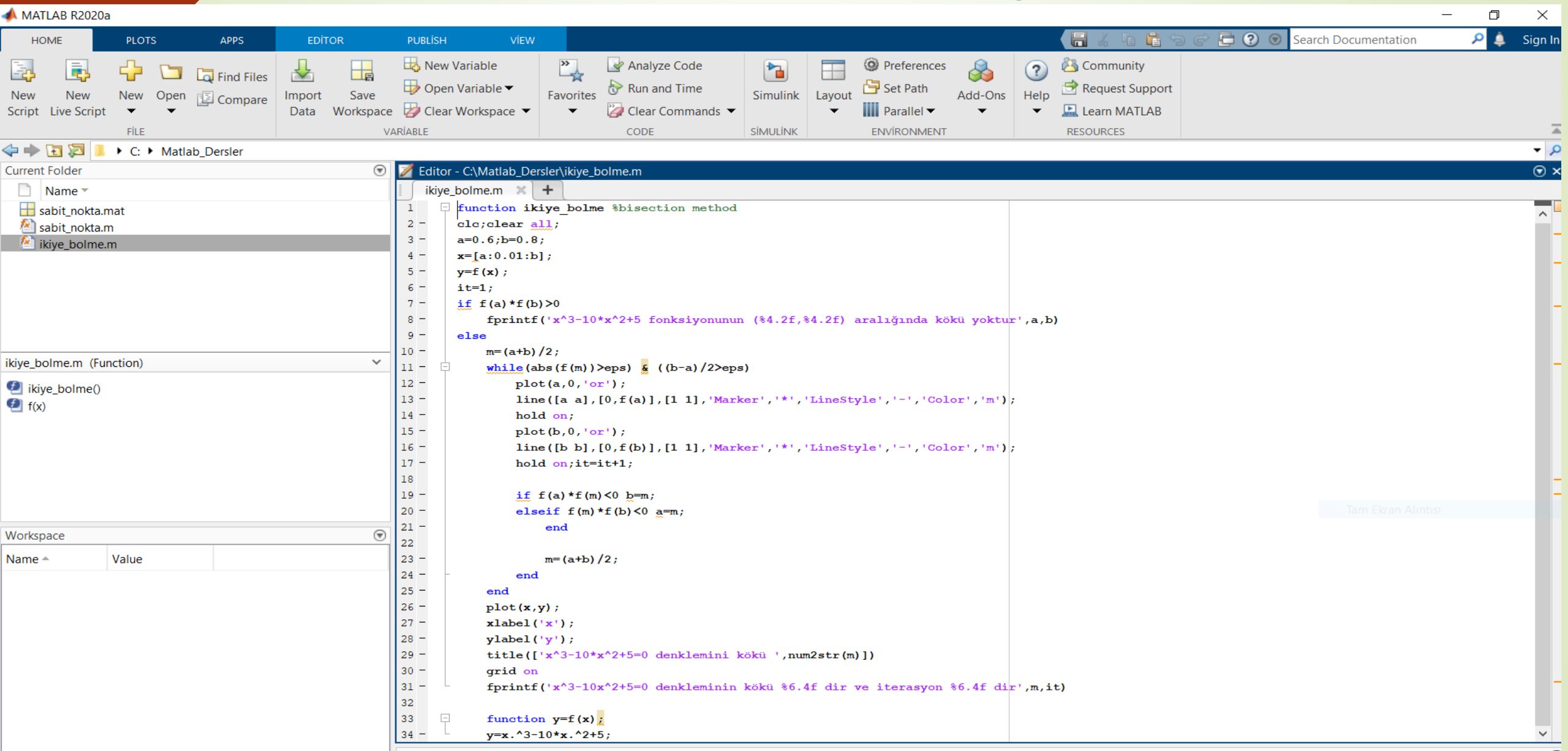
2.Adım: m=(a+b)/2

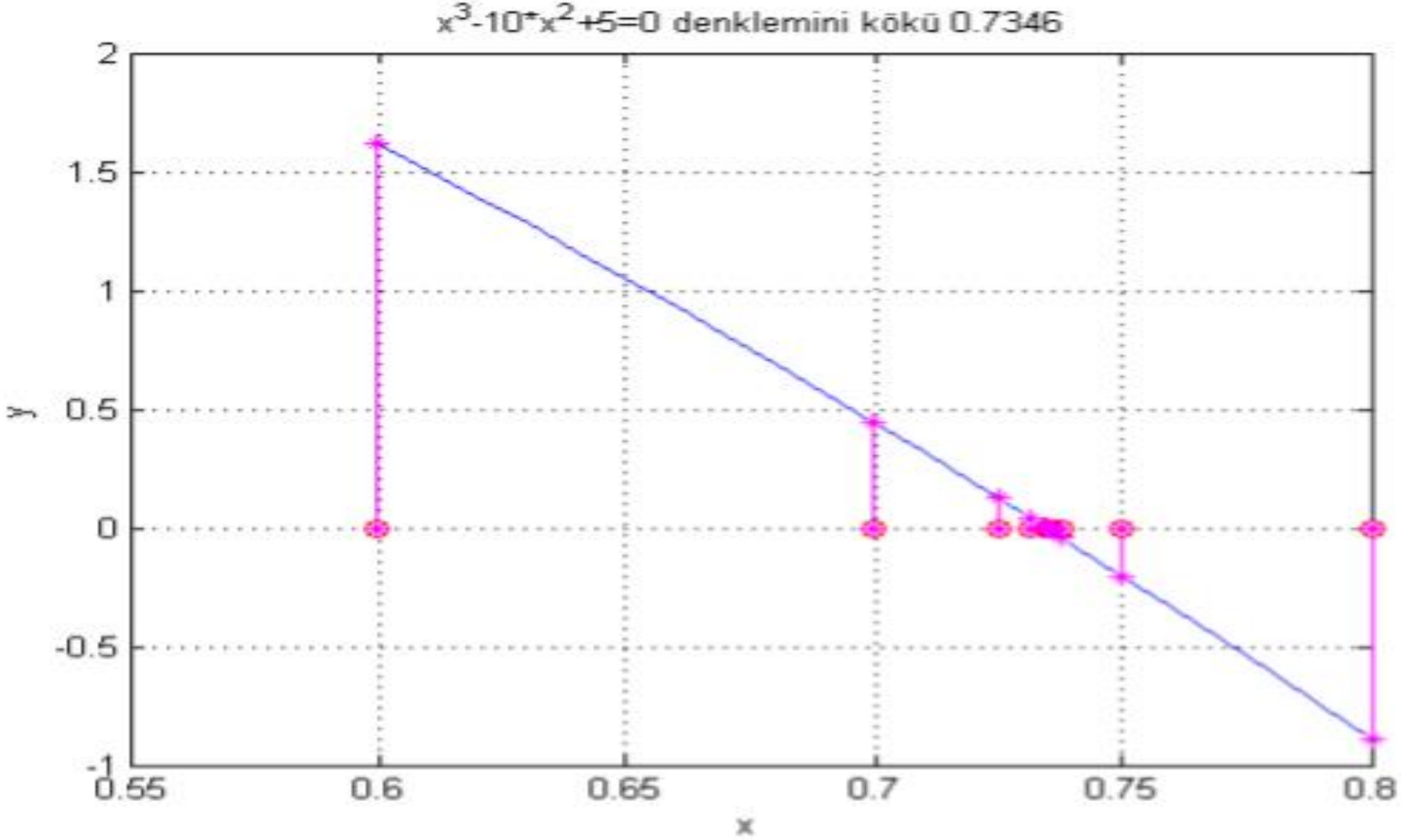
3.Adım: Eğer f(m)<eps veya |b-a|/2<eps ise m çözümdür ve programdan çık

4.Adım: Eğer f(a)f(m)<0 ise b➔m olarak belirle, Eğer f(m)f(b)<0 ise a➔m olarak belirle

5.Adım i➔i+1 olarak belirle ve 2.Adıma dön

Örnek: f(x)=x^3-10x^2+5=0 fonksiyonunun (0.6,0.8) aralığındaki kökünü ikiye bölme yöntemi ile MATLAB programında yazınız.




x^3-10*x^2+5=0 denklemini kökü 0.7346
2
1.5
1
0.5
0
-0.5
-1
y
0.55
0.6
0.65
0.7
0.75
0.8
x

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

4.DERS (GECE GRUBU)

Arş. Gör. Alev KAYA

26.03.2021
SAAT:16:00-17:00

Lineer Olmayan Denklem Sistemlerimin Çözümü

- Kapalı Yöntemler :

A-Regula Falsi yöntemi

- Açık Yöntemler :

A-Fixed point Iteration yöntemi

- LAB: Regula Falsi yöntemi ve Fixed Point yöntemi Matlab örnek programı

Lineer Olmayan Denklem Sistemlerimin Çözümü

- Kapalı Yöntemler :

A-Regula Falsi yöntemi (Yer Değiştirme Methodu)

- İkiye bölme yöntemi, köklerin bulunması için geçerli bir yöntem olmasına rağmen **'kaba kuvvet'** yaklaşımı oldukça verimsiz,

- İkiye bölme yönteminin eksiklerinden biri, xüst ile xalt arasında kalan aralığı ikiye bölerken f(xalt) ve f(xüst) değerlerini göz önüne almamasıdır.

- **Örneğin;** f(xalt) sıfıra f(xüst) ' den daha yakınsa, kökün xalt değerine ,xüst değerinden daha yakın olma olasılığı mevcuttur.

Lineer Olmayan Denklem Sistemlerimin Çözümü

- **Kapalı Yöntemler :**

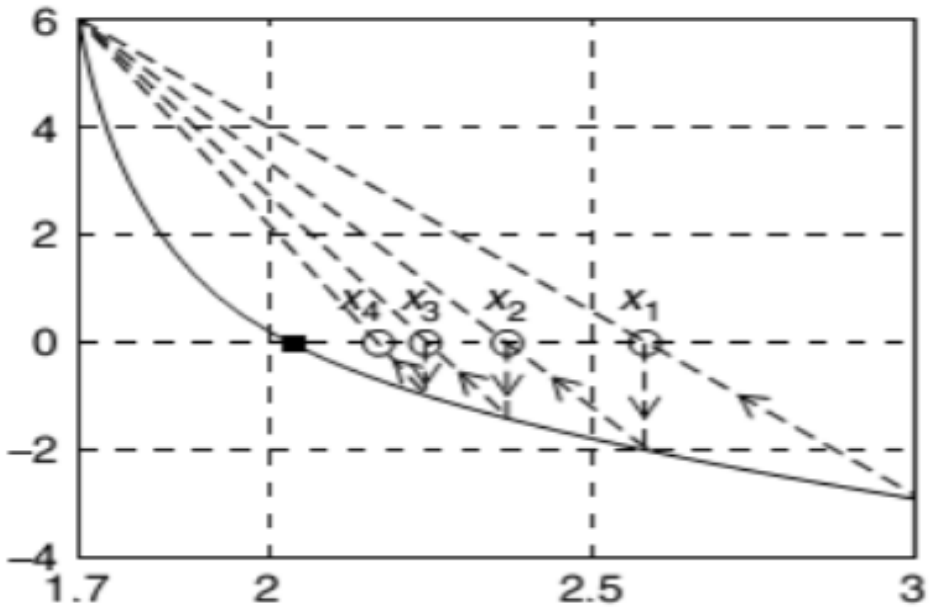
A-Regula Falsi yöntemi (Yer Değiştirme Methodu)

- Yer değiştirme formülü kullanılarak hesaplanan xtahmin değeri daha sonra lineer olmayan fonksiyon f(xtahmin) ile aynı işaretli yapan xalt ve xüst değeri ile yer değiştirilir.
- Böylece xalt ve xüst değerleri her zaman gerçek kökü kıskaca alır.
- Bu yöntem, kapalı yöntemler arasında tercih edilen bir çözüm yaklaşımıdır.

$$x_{tahmin} = x_{üst} - \frac{f(x_{üst})(x_{alt} - x_{üst})}{f(x_{alt}) - f(x_{üst})}$$

REGULA FALSİ YÖNTEMİ (REGULA FALSI – FALSE POSITION METHOD)

Eğer f(x)=0 denkleminin (a,b) aralığında kökü olması için f(a).f(b)<0 koşulu sağlanması gerekmektedir.



Yöntem için adımlar aşağıdaki gibidir:

Algoritma

1.Adım: i=1 olarak belirle

$$x^{\star} = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

2.Adım:

3.Adım: Eğer $f(x^*)<eps$ veya $|b-a|/2<eps$ ise x^* çözümdür ve programdan çık

4.Adım: Eğer $f(a)f(x^*)<0$ ise b→x^* olarak belirle, Eğer $f(x^*)f(b)<0$ ise a→x^* olarak belirle

5.Adım i→i+1 olarak belirle ve 2.Adıma dön

Örnek: tan(π-x)-x=0 fonksiyonunun (1.6,3) aralığındaki kökünü Regula Falsi ile MATLAB programında yazınız.

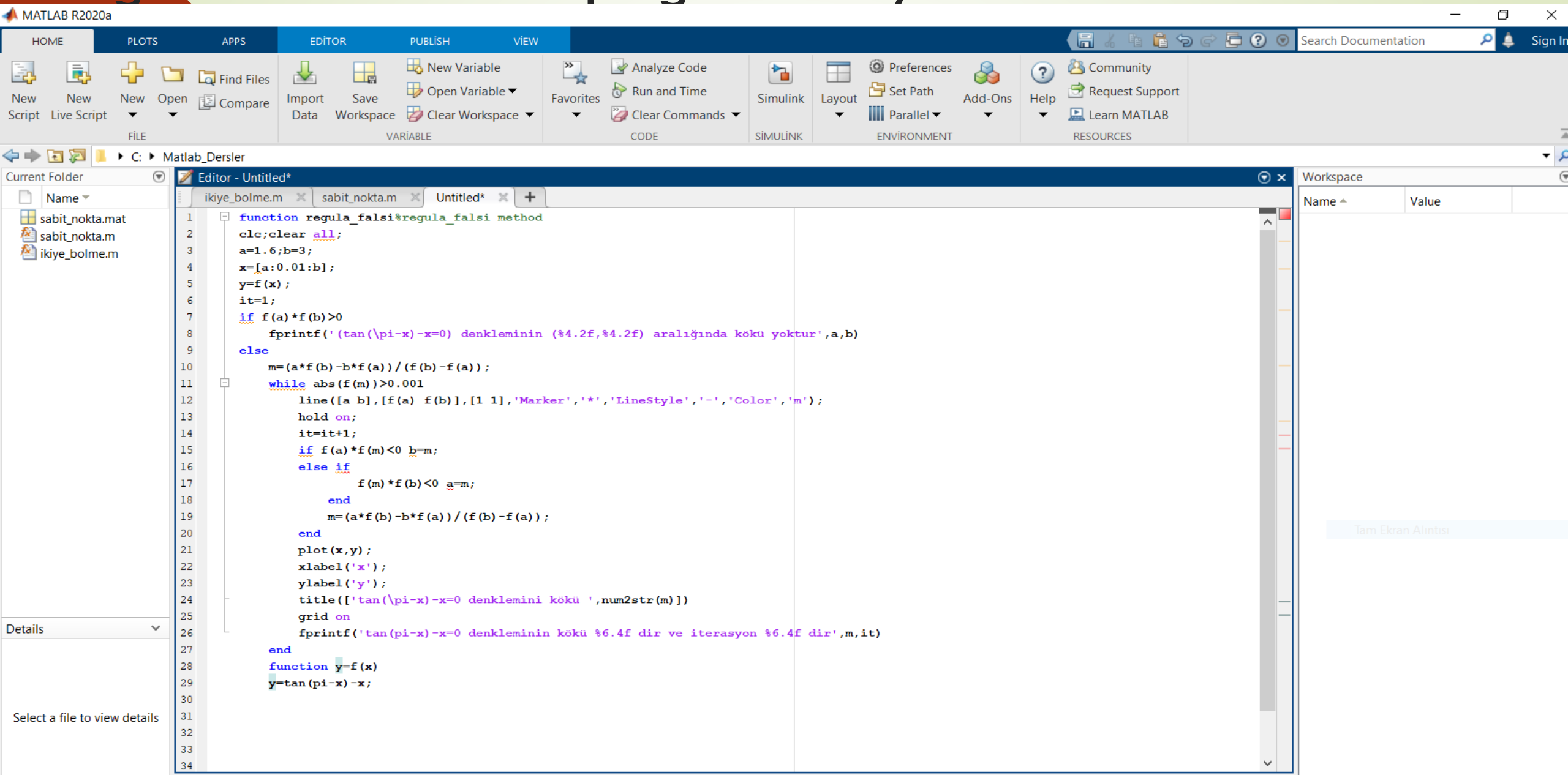
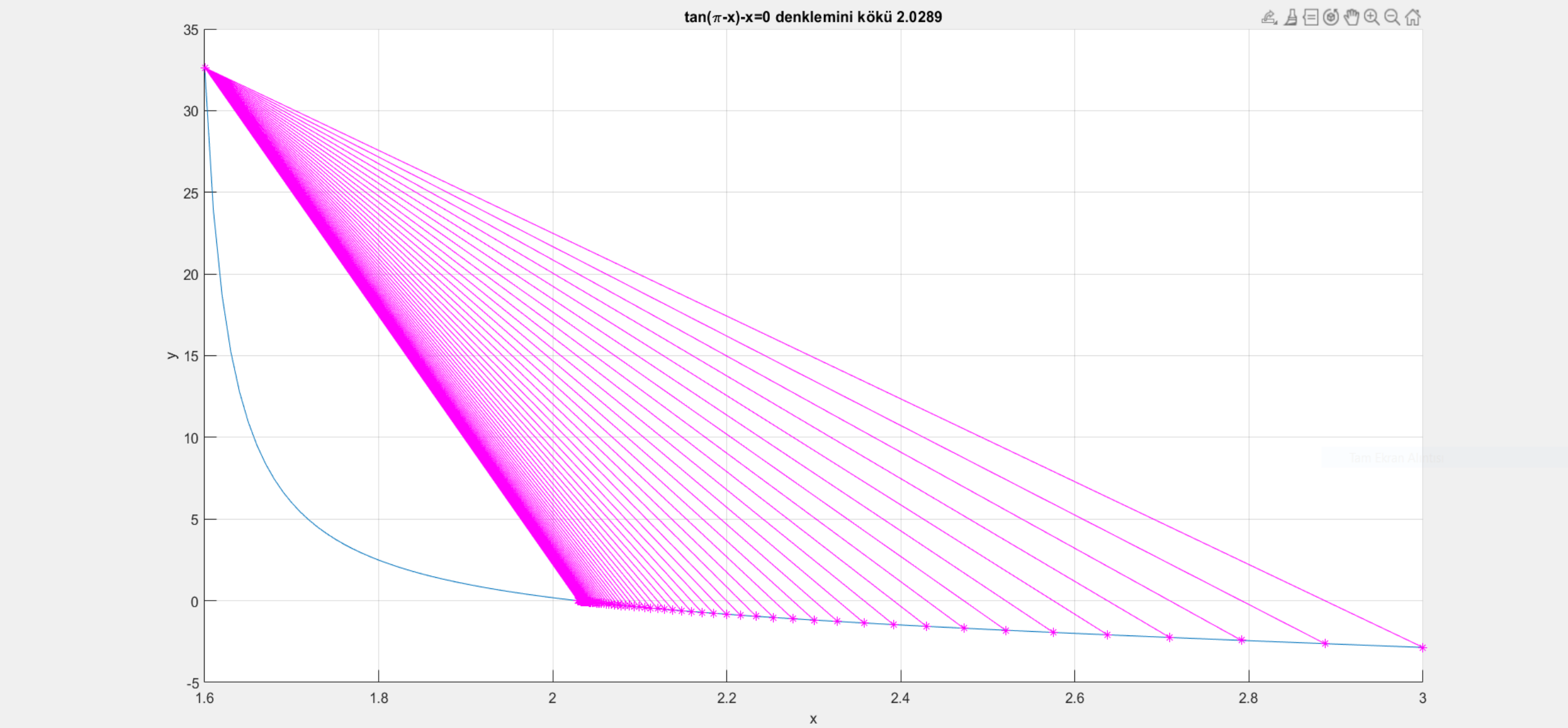


Figure 1

File Edit View Insert Tools Desktop Window Help

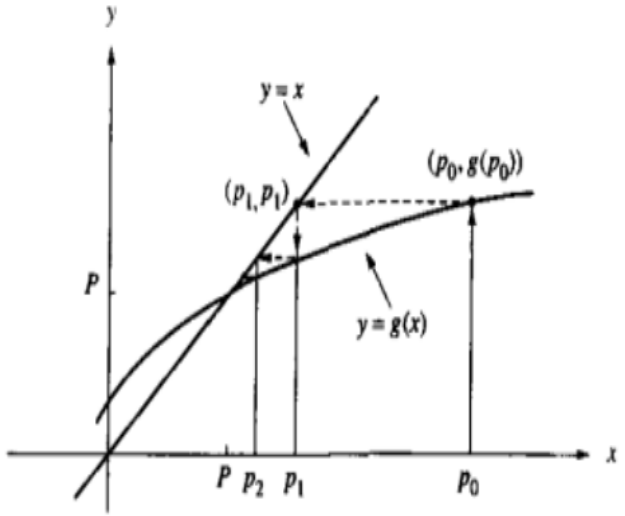


BÖLÜM 3 - F(X)=0 FORMUNDA LİNEER OLMAYAN DENKLEMLERİN ÇÖZÜMLERİ

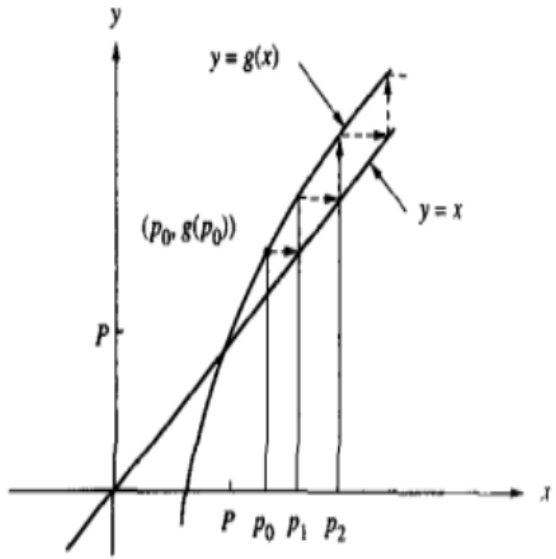
f(x)=0 denkleminin çözümüne, denklemin kökleri veya fonksiyonun sıfırları denir. Bir denklemin bir yada birden fazla kökü olabildiği gibi hiç kökü de yoktur. Örneğin $\sin x - x = 0$ denkleminin bir tek x=0 kökü varken, $\tan x - x = 0$ denkleminin $x = 0, \pm 4.493, \pm 7.725, \ldots$ şeklinde sonsuz kökü vardır. Bu bölümde denklemin kökünün bulunması için yöntemler verilip MATLAB kodu yazılacaktır.

SABİT NOKTA İTERASYONU (FIXED POINT ITERATION)

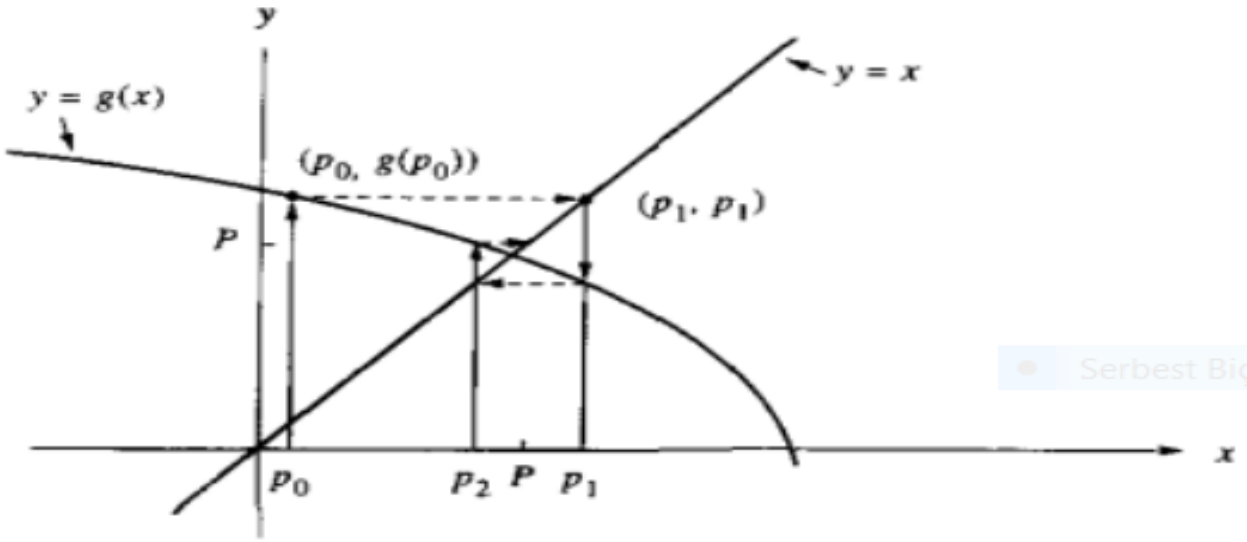
Eğer g(x) fonksiyonu ve g'(x) fonksiyonu sürekli ve |g'(x)|<1 ise x=g(x) denkleminin bir tek çözümü vardır ve çözüm $x_{k+1} = g(x_k)$ iterasyon yöntemi ile elde edilir.



➔Yakınsaklık: $0 < g'(P) < 1$. durumu

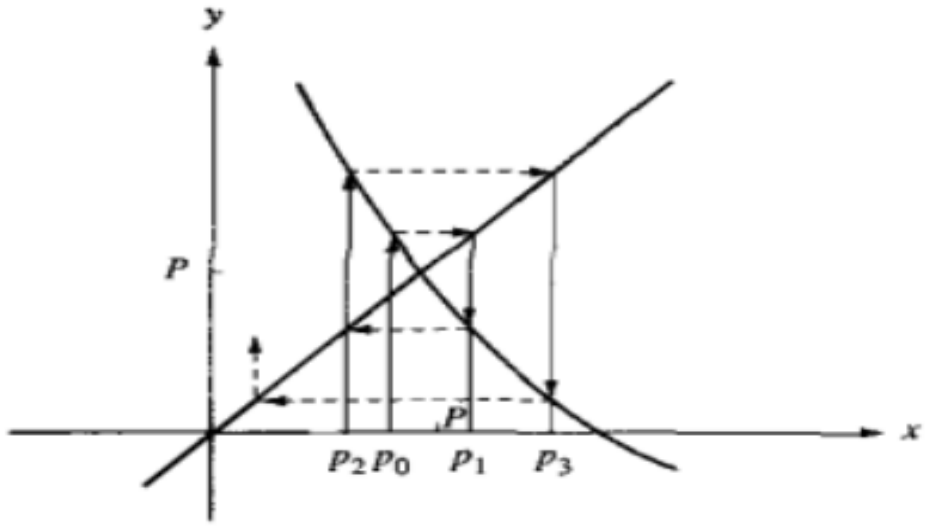


➔ Iraksak: $1 < g'(P)$.



➔Yakınsaklık

$n-1 < g'(P) < 0.$ durumu



➔Iraksaklık: $g'(P) < -1.$ durumu

Alıştırma 1. $x = 2^{-x}$ denkleminin

Örnek: x=2^-x denkleminin kökünü [0,1] aralığında bulacak şekilde sabit nokta iterasyonunu MATLAB kodunu yazınız.

MATLAB R2020a

HOME PLOTS APPS EDİTOR PUBLİSH VİEW

Search Documentation Sign In

New Script | New Live Script | New | Open | Find Files | Compare — FİLE
Import Data | Save Workspace | New Variable | Open Variable | Clear Workspace — VARİABLE
Favorites | Analyze Code | Run and Time | Clear Commands — CODE
Simulink — SİMULİNK
Layout | Preferences | Set Path | Parallel | Add-Ons — ENVİRONMENT
Help | Community | Request Support | Learn MATLAB — RESOURCES

C: ▸ Matlab_Dersler

Current Folder

Name
- sabit_nokta.mat
- sabit_nokta.m
- ikiye_bolme.m

ikiye_bolme.m (Function)
- ikiye_bolme()
- f(x)

Workspace

Name | Value

Editor - C:\Matlab_Dersler\sabit_nokta.m

sabit_nokta.m

```
1   function sabit_nokta
2   clc;clear all;
3   %x=2^(-x), [0,1]
4   xx=linspace(0,1,100);
5   gx=2.^(-xx);
6   x0=0;
7   plot(x0,0,'or');hold on;
8   x1=2^(-x0);
9   plot(x1,0,'or');
10  line([x0 x0],[0,x1],[1 1],'Marker','*','LineStyle','-','Color','m');
11  hold on;
12  it=1;
13  while abs(x1-x0)>eps
14      x0=x1;
15      x1=2^(-x0);
16      plot(x1,0,'or');
17      line([x0 x0],[0,x1],[1 1],'Marker','*','LineStyle','-','Color','m');
18      hold on;
19      it=it+1;
20  end
21  plot(xx,xx,xx,gx);
22  xlabel('x');
23  ylabel('y');
24  title(['g(x)=2^{-x} eğrisi ile f(x)=x doğrusunun kesimnoktası ',num2str(x1)])
25  fprintf('g(x)=2^{-x} eğrisi ile f(x)=x doğrusunun kesim noktası %6.4f dir ve iterasyon %6.4f dir',x1,it)
26  save ('sabit_nokta.mat','it','x1','x0')
```

Command Window

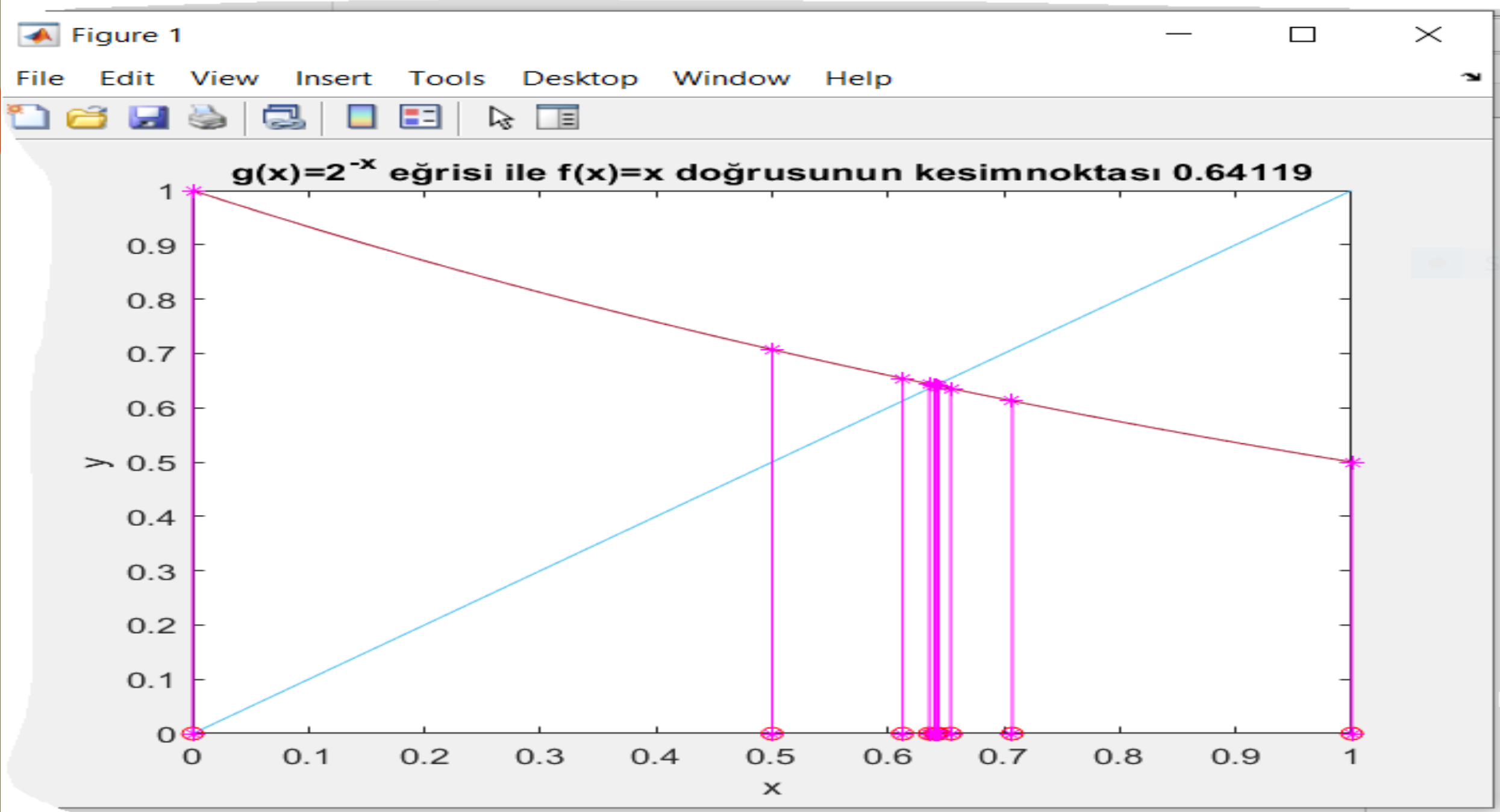


Figure 1
File Edit View Insert Tools Desktop Window Help
g(x)=2^-x eğrisi ile f(x)=x doğrusunun kesimnoktası 0.64119
y
x
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

5.DERS

Arş. Gör. Alev KAYA

02.04.2021
SAAT:16:00-17:00

Lineer Olmayan Denklem Sistemlerimin Çözümü

- Açık Yöntemler:
- A-Newton Raphson Yöntemi
- B-Sekant Yöntemi
- LAB: Newton Raphson yöntemi ve Sekant yöntemi Matlab örnek programı

Lineer Olmayan Denklem Sistemlerimin Çözümü
Açık Yöntemler

NEWTON – RHAPSON YÖNTEMİ (NEWTON-RHAPSON METHOD)

Kök bulma algoritmalarından en ünlüsü Newton-Rhapson yöntemidir. Çözüme çok hızlı yakınsayan basit bir programdır. Bu yöntem hem fonksiyonun kendisini hemde türevini içerir. Yöntem aşağıdaki iterasyon ile sonraki noktayı bulmayı hedefler:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Yöntem için adımlar aşağıdaki gibidir:

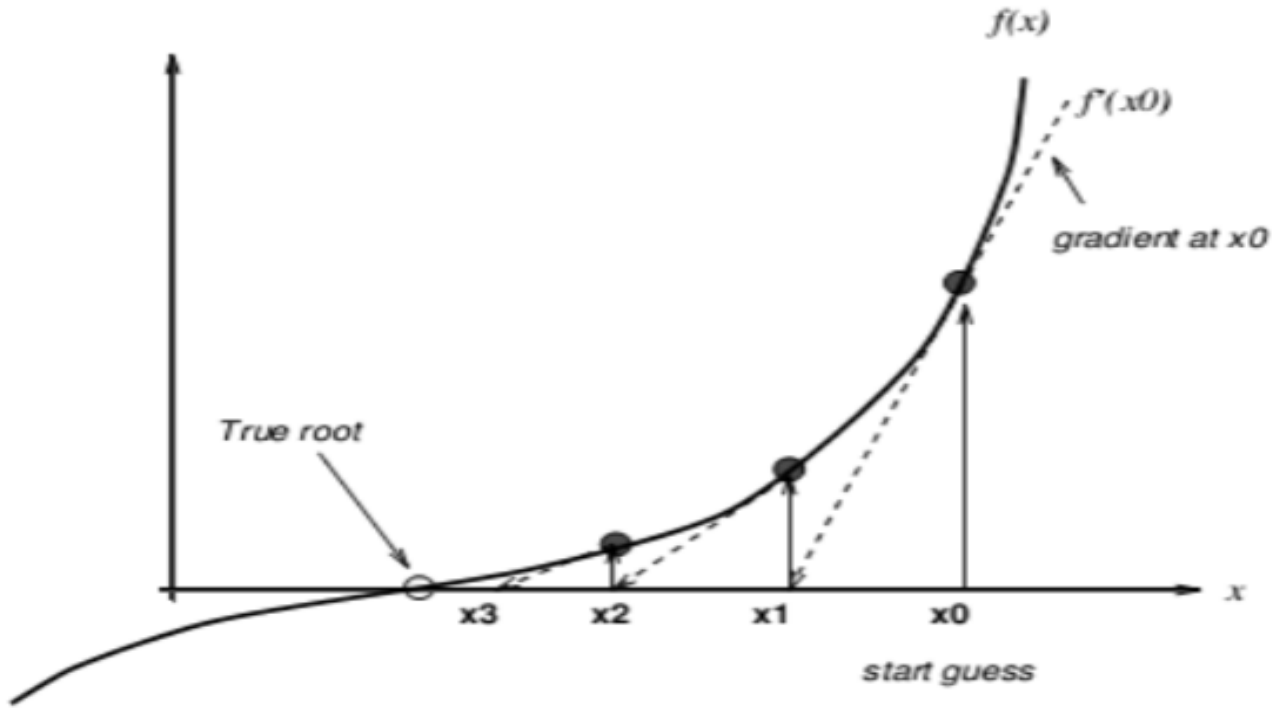
1.Adım: i=1 olarak belirle. Eğer $f'(x_i) = 0$ ise programı durdur ve hata mesajı ver.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

2.Adım:

3.Adım: Eğer $f(x_{i+1})$<eps ise x_{i+1} çözümdür ve programdan çık

4.Adım i➔i+1 olarak belirle ve 2.Adıma dön



Örnek: tan(π-x)-x=0 fonksiyonunun (1.6,3) aralığındaki kökünü Newton-Rhapson yöntemi ile MATLAB programında yazınız.

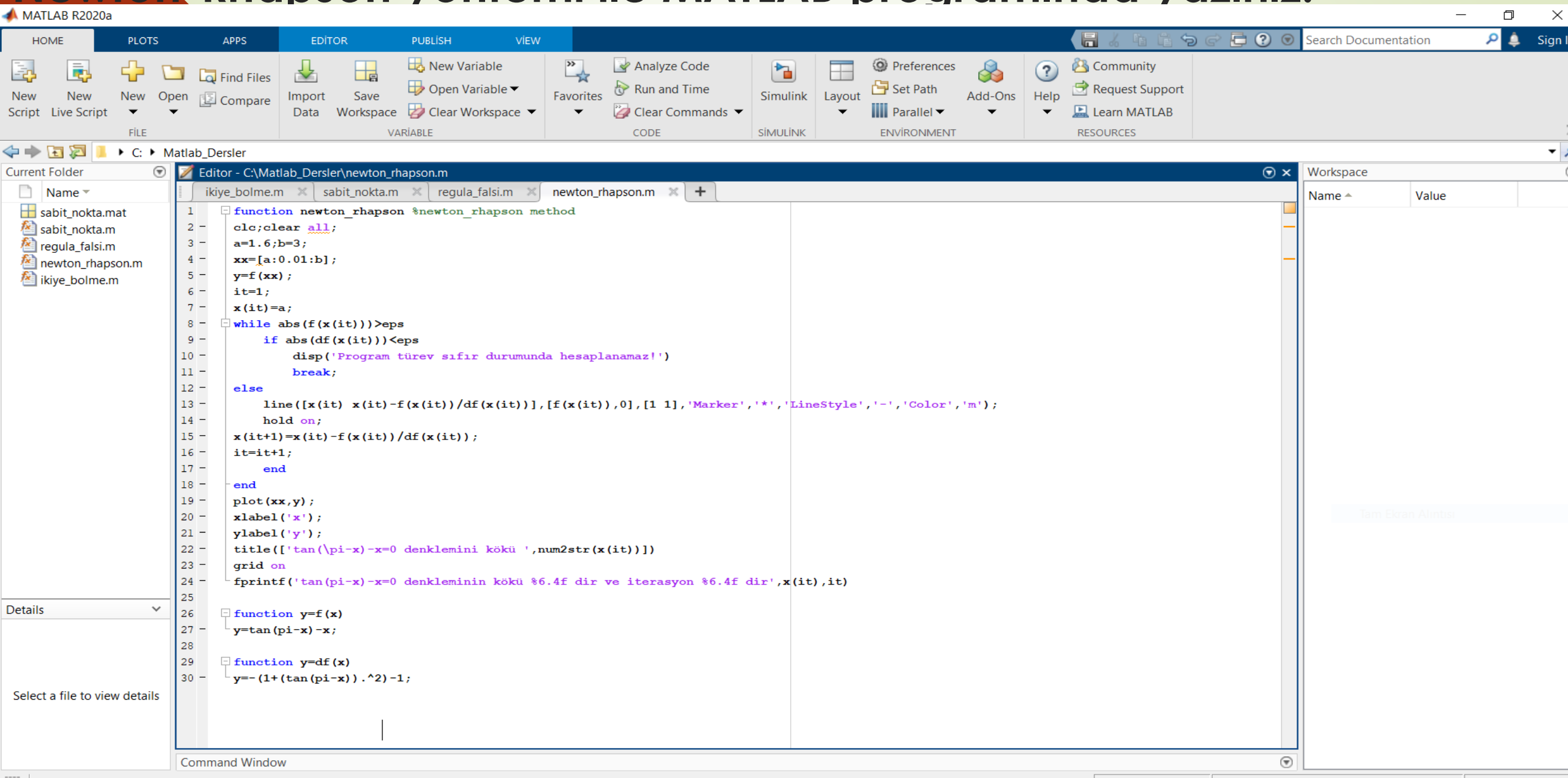
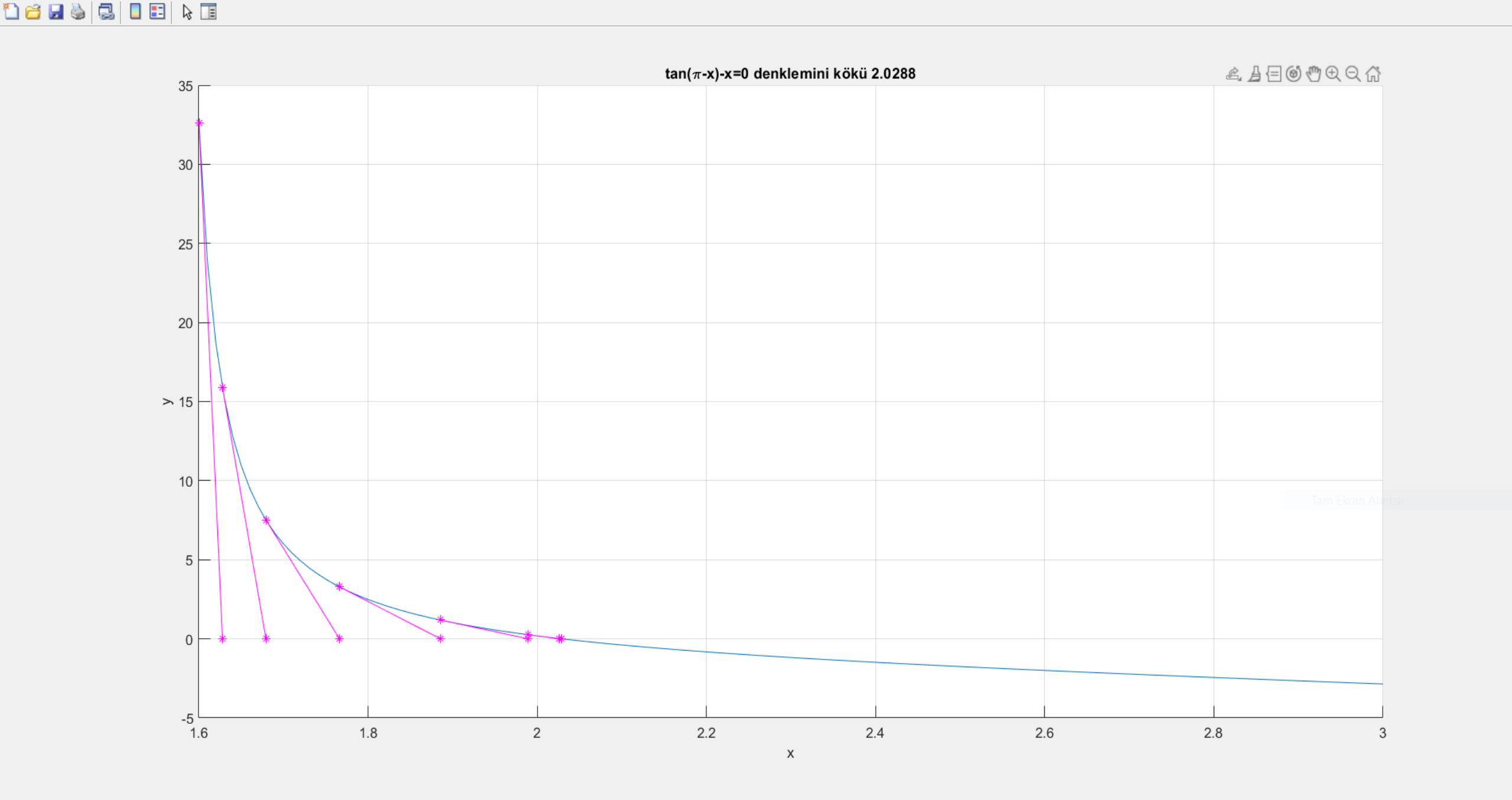


Figure 1

File Edit View Insert Tools Desktop Window Help



Lineer Olmayan Denklem Sistemlerimin Çözümü
Açık Yöntemler

KİRİŞLER YÖNTEMİ (SECANT METHOD)

Kirişler yöntemi Newton-Rhapson yöntemindeki türevin sonlu fark yaklaşımı yazılması ile elde edilir. Newton rhapson yönteminde türev yerine

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

yazılarak iterasyonu

$$x_{k+1} \leftarrow x_k - \frac{f(x_k)}{\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}} \quad \rightarrow \quad x_{k+1} \leftarrow x_k - \frac{(x_k - x_{k-1})f(x_k)}{f(x_k) - f(x_{k-1})}$$

$$x_{k+1} \leftarrow \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Olarak yazarız.

Yöntem için adımlar aşağıdaki gibidir:

Algoritma

1.Adım: i=1 olarak belirle.

2.Adım: $$x_{k+1} \leftarrow \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

3.Adım: Eğer $f(x_{i+1})<eps$ ise x_{i+1} çözümdür ve programdan çık

4.Adım i➔i+1 olarak belirle ve 2.Adıma dön

Örnek: f(x)=x^3-3x+2 fonksiyonunun x0=-2.6, x1=-2.4 vererek kökünü Kirişler yöntemi ile MATLAB programında yazınız.

MATLAB R2020a

HOME PLOTS APPS EDİTOR PUBLİSH VİEW

New Open Save Find Files Compare Print — FİLE
Go To Find — NAVİGATE
Insert Comment Indent — EDİT
Breakpoints — BREAKPOİNTS
Run Run and Advance Run Section Advance Run and Time — RUN

C: ▸ Matlab_Dersler

Current Folder

Name
- sabit_nokta.mat
- sabit_nokta.m
- regula_falsi.m
- newton_rhapson.m
- kirisler_yontemi.m
- ikiye_bolme.m

Details

Select a file to view details

Editor - C:\Matlab_Dersler\kirisler_yontemi.m

ikiye_bolme.m | sabit_nokta.m | regula_falsi.m | newton_rhapson.m | kirisler_yontemi.m

```
1  function kirisler_yontemi
2  clc;clear all;
3  it=1;
4  x(it)=-2.6;
5  it=it+1;
6  x(it)=-2.4;
7  xx=[x(1):0.01:-1.98];
8  y=f(xx);
9  while abs(f(x(it)))>0.001
10     x(it+1)=(x(it-1)*f(x(it))-x(it)*f(x(it-1)))/(f(x(it))-f(x(it-1)));
11     line([x(it-1) x(it+1)],[f(x(it-1)),f(x(it+1))],[1 1],'Marker','*','LineStyle','-','Color','m');
12     hold on;
13     it=it+1;
14 end
15 plot(xx,y);
16 xlabel('x');
17 ylabel('y');
18 title(['x.^3-3*x+2=0 denklemini kökü ',num2str(x(it))])
19 grid on
20 fprintf('x.^3-3*x+2=0 denkleminin kökü %6.4f dir ve iterasyon %6.4f dir',x(it),it)
21
22 function y=f(x);
23 y=x.^3-3*x+2;
```

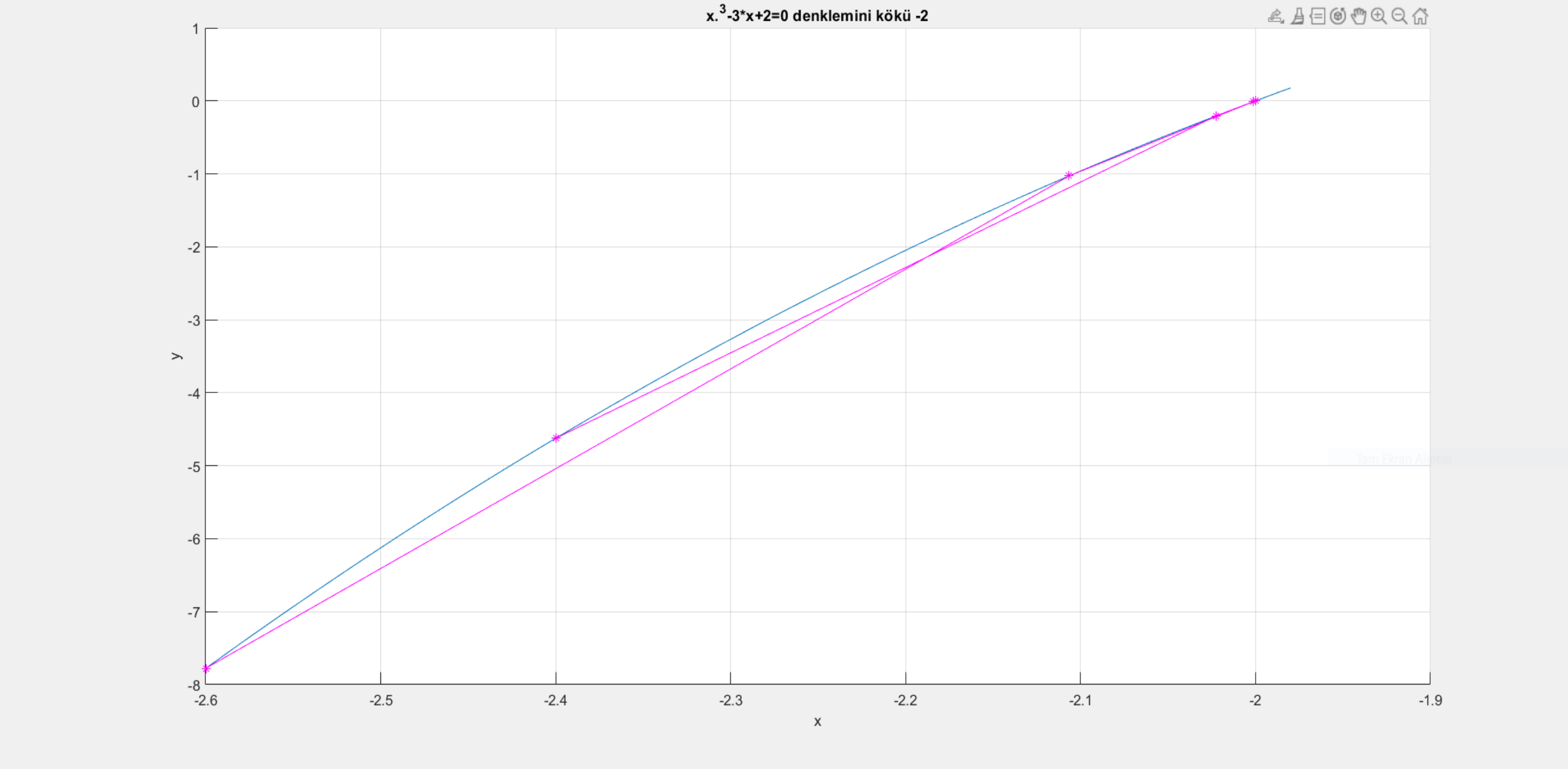
Workspace

Name | Value

Command Window

Figure 1

File Edit View Insert Tools Desktop Window Help



2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

6.DERS

Arş. Gör. Alev KAYA

09.04.2021
SAAT:09:00-10:00

Lineer Denklem Sistemlerinin Çözümü

- Doğrudan Yöntemler
- A-Cramer Yöntemi
- B-Gauss Jordan Yöntemi
- LAB: Cramer yöntemi Matlab örnek programı

Lineer Denklem Sistemi

Belirli sayıda bilinmeyen ve belli sayıda denklemden oluşan bir denklem sistemi lineer terimlerden oluşuyorsa bu sistem lineer denklem sistemi olarak adlandırılır.

$$2x - 3y = 5$$
$$-2x + y = -1$$

denklem sistemi iki bilinmeyen içeren lineer bir denklem sistemidir. Genel olarak n adet bilinmeyen $(x_1, x_2, x_3, \ldots, x_n)$ içeren lineer bir denklem sistemi aşağıda gösterildiği gibi çık halde veya daha basit olarak matris formunda yazılabilir.

$$\left.\begin{array}{lllll}
a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + & \ldots\ldots\ldots & a_{1n}x_n = b_1 \\
a_{21}x_1 + & a_{22}x_2 + & a_{23}x_3 + & \ldots\ldots\ldots & a_{2n}x_n = b_2 \\
a_{31}x_1 + & a_{32}x_2 + & a_{33}x_3 + & \ldots\ldots\ldots & a_{3n}x_n = b_3 \\
\ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots \\
a_{n1}x_1 + & a_{n2}x_2 + & a_{n3}x_3 + & \ldots\ldots\ldots & a_{nn}x_n = b_n
\end{array}\right\} \quad A.\vec{x} = \vec{b}$$

Lineer Denklem Sistemi

$$
\begin{aligned}
a_{11}x_1 + \quad & a_{12}x_2 + \quad a_{13}x_3 + \quad \ldots\ldots\ldots \quad a_{1n}x_n = \quad b_1 \\
a_{21}x_1 + \quad & a_{22}x_2 + \quad a_{23}x_3 + \quad \ldots\ldots\ldots \quad a_{2n}x_n = \quad b_2 \\
a_{31}x_1 + \quad & a_{32}x_2 + \quad a_{33}x_3 + \quad \ldots\ldots\ldots \quad a_{3n}x_n = \quad b_3 \\
\ldots\ldots\ldots \quad & \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \ldots\ldots\ldots \quad \ldots\ldots \\
a_{m1}x_1 + \quad & a_{m2}x_2 + \quad a_{m3}x_3 + \quad \ldots\ldots\ldots \quad a_{mn}x_n = \quad b_m
\end{aligned}
$$

$$
\underbrace{\begin{bmatrix} a_{11} & a_{12} & \ldots. & a_{1n} \\ a_{21} & a_{22} & \ldots. & a_{2n} \\ \ldots. & \ldots. & \ldots. & \ldots. \\ a_{m1} & a_{m2} & \ldots. & a_{mn} \end{bmatrix}_{m\times n}}_{A} \times \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \ldots. \\ x_n \end{bmatrix}_{n\times 1}}_{\vec{x}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \ldots. \\ b_m \end{bmatrix}_{m\times 1}}_{\vec{b}}
$$

A: katsayılar matrisi

x: bilinmeyen vektör

b: sağ taraf vektörü

Örnek

$$\begin{bmatrix} 3 & 2 & 4 \\ 1 & -2 & 0 \\ -1 & 3 & 2 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 14 \\ 7 \\ 2 \end{bmatrix}$$

$$\begin{aligned} 3x_1 + 2x_2 + 4x_3 &= 14 \\ x_1 - 2x_2 + 0x_3 &= 7 \\ -x_1 + 3x_2 + 4x_3 &= 2 \end{aligned}$$

Lineer Denklem Sistemlerinin Çözümü

Lineer bir denklem sisteminin çözülerek bilinmeyen x_i değerlerinin bulunmasında değişik yöntemler kullanılır. Bu yöntemler 2 grup halinde ayrılabilir.

1-) Analitik (Direkt) Yöntemler: Denklem sisteminin çözümünü matematik anlamda tam olarak veren yöntemlerdir. Bu yöntemler sayesinde doğrudan aranan çözüm elde edilir.

- Matris Tersi Yöntemi
- Cramer Yöntemi
- Eliminasyon Yöntemi
- Gauss Eliminasyon Yöntemi
- Gauss-Jordan Yöntemi
- LU Ayırma Yöntemi

Lineer Denklem Sistemlerinin Çözümü

2-) İteratif (Dolaylı) Yöntemler: Çözümü bulmak için öncelikle tahmini değerlerden başlanır ve adım adım ardışık hesaplamalarla belirli tolerans sınırları içinde aranan çözüme ulaşılır.

- Basit İterasyon (Jacobi) Yöntemi
- Gauss-Seidel Yöntemi
- Rölaksasyon (SOR) Yöntemi

Matrisin Tersi ile Çözümleme

Verilen $A.x=b$ denklem sisteminde, katsayılar matrisinin tersi (A^{-1}) hesaplandığında çözüm vektörü iki matrisin çarpımından elde edilebilir.

$$A.\vec{x} = \vec{b} \qquad \Rightarrow \qquad \vec{x} = A^{-1}.\vec{b}$$

Matris tersinin hesabı için farklı yöntemler vardır. Fakat bu yöntemlerde, eleman sayısı ne kadar fazla ise matris tersini bulmak için daha fazla bilgisayar hafızasına ve hesaplama zamanına ihtiyaç duyulur.

Matrisin Tersi Nasıl Bulunur

Bir matrisin tersini alabilmek için:

- **Matrisin determinantını ve matrisin ekini (adjoint) bulmak gerekir.**

Bir matrisin ekinin bulunması için:

1. **Matrisin bütün elemanlarının eş-çarpanları bulunur.**
2. **Bulunan eş-çarpanlardan yeni bir matris oluşturulur.**
3. **Bu matrisin transpozesi alınır.**

Örnek

$A = \begin{bmatrix} 1 & 4 & 2 \\ 3 & 3 & 1 \\ 2 & 0 & 1 \end{bmatrix}$ matrisinin tersini bulunuz.

1) Matrisin determinantı bulunur.

$$|A| = -13$$

2) Matrisin eş-çarpanları bulunur.

$$A_{11} = (-1)^{1+1}.\begin{vmatrix} 3 & 1 \\ 0 & 1 \end{vmatrix} = 1 \times 3 = 3$$

$$A_{12} = (-1)^{1+2}.\begin{vmatrix} 3 & 1 \\ 2 & 1 \end{vmatrix} = (-1)(3-2) = -1$$

$$A_{13} = (-1)^{1+3}.\begin{vmatrix} 3 & 3 \\ 2 & 0 \end{vmatrix} = 0-6 = -6$$

$$A_{21} = (-1)^{2+1}.\begin{vmatrix} 4 & 2 \\ 0 & 1 \end{vmatrix} = (-1)(4-0) = -4$$

$$A_{22} = (-1)^{2+2}.\begin{vmatrix} 1 & 2 \\ 2 & 1 \end{vmatrix} = 1-4 = -3$$

$$A_{23} = (-1)^{2+3}.\begin{vmatrix} 1 & 4 \\ 2 & 0 \end{vmatrix} = (-1)(0-8) = 8$$

$$A_{31} = (-1)^{3+1}.\begin{vmatrix} 4 & 2 \\ 3 & 1 \end{vmatrix} = 4-6 = -2$$

$$A_{32} = (-1)^{3+2}.\begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix} = (-1)(1-6) = 5$$

$$A_{33} = (-1)^{3+3}.\begin{vmatrix} 1 & 4 \\ 3 & 3 \end{vmatrix} = 3-12 = -9$$

Örnek-Devam

3) Ek-matris bulunur.

$$Ek(A)=\begin{bmatrix} 3 & -1 & -6 \\ -4 & -3 & 8 \\ -2 & 5 & -9 \end{bmatrix}^{T}=\begin{bmatrix} 3 & -4 & -2 \\ -1 & -3 & 5 \\ -6 & 8 & -9 \end{bmatrix}$$

4) Matrisin tersi bulunur.

$$A^{-1}=\frac{1}{|A|}.Ek(A)=\frac{1}{-13}\begin{bmatrix} 3 & -4 & -2 \\ -1 & -3 & 5 \\ -6 & 8 & -9 \end{bmatrix}$$

MATLAB'da Matrisin Tersini Alma

Matrisin tersini verir.

inv (matris)

tersi hesaplanacak matris

Örnek:

$$A = \begin{bmatrix} 1 & 4 & 2 \\ 3 & 3 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

```
>> A=[1 4 2;3 3 1;2 0 1]

A =

     1     4     2
     3     3     1
     2     0     1

>> inv(A)

ans =

   -0.2308    0.3077    0.1538
    0.0769    0.2308   -0.3846
    0.4615   -0.6154    0.6923

>>
```

Lineer Denklem Sistemlerinin Çözümü

Lineer denklem sistemlerinin Matlab Programı ile çözümü 3 ana başlıkta incelenebilir.

1. Cramer Metodu
2. Matris Tersi Yöntemi
3. "\" Operatörü Yöntemi

Denklem Sayısı ve Bilinmeyen Sayısı Eşit Olan Denklemler

- Bu tip lineer denklemlerin oluşturdukları katsayılar matrisi KARE MATRİS olacaktır.
- Bu tip Denklem sayısı ve bilinmeyen sayısı eşit olan denklem takımlarının çözümün de Cramer Metodu, Matris İnversi yöntemi yada "\" operatörü ile çözüm yöntemi kullanılabilir.
- Cramer yöntemi, denklem sayısı ile bilinmeyen sayısının eşit olması durumunda, katsayılar matrisinin determinantı sıfırdan farklı ise uygulanır.

Cramer Yöntemi ile Çözümleme

Klasik yöntemlerden biri olup, çözüm iki matrisin determinantları oranı olarak elde edilir. Bu yöntemde, *n* tane bilinmeyen içeren

$$A.\vec{x} = \vec{b}$$

şeklindeki lineer denklem sisteminin çözümü;

$$x_i = \frac{|D_i|}{|A|} \qquad (i = 1,2,3,\ldots,n)$$

$|D_i|$: Katsayılar matrisinde (A), *i. sütun* atılıp yerine b vektörünün konması ile elde edilen matrisin determinantıdır.

Bu yöntemde, her biri (n×n) boyutundaki (n+1) adet matrisin determinantının bulunup, bunların oranlanması gerekir. Bundan dolayı işlem sayısı fazla ve çözüm süresi uzundur.

Matrislerde Determinant (2×2)

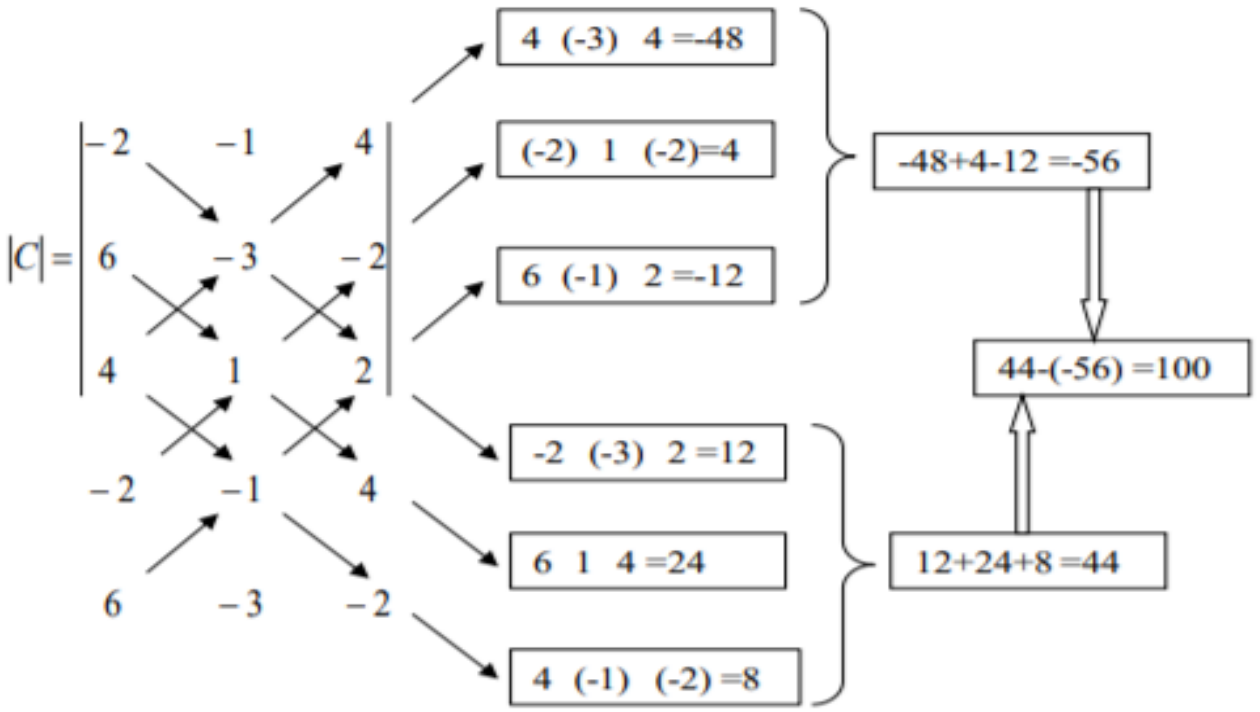
Matrisin köşegenindeki elemanların çarpımından ters köşegenindeki elemanlarının çarpımı birbirinden çıkarılarak bulunur.

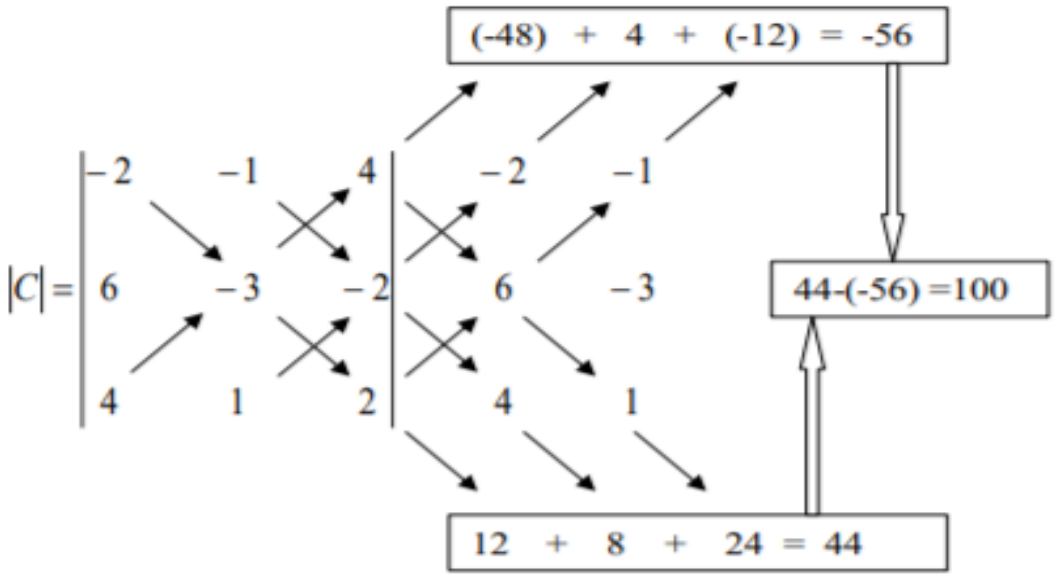
$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}_{2\times 2}$$

$$|A| = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = (a_{11} \times a_{22}) - (a_{12} \times a_{21})$$

Matrislerde Determinant (3×3)

1. ve 2. satırlar/sütunlar, aynı matrise 4. ve 5. satır/sütun olarak eklenir ve oluşan matriste sol köşegen çarpımlarının toplamından sağ köşegen çarpımlarının toplamı çıkarılarak determinant bulunur.





Seçilen Satır/Sütuna göre Determinant Bulma

Minör ve Kofaktör kullanılarak determinant hesaplanır.

- **Bir kare matrisin bulunduğu a_{ij} elemanının i'nci satır ve j'nci sütunu atıldığında geriye kalan M_{ij} matrisinin determinantına a_{ij} elemanının küçüğü (minörü) denir.**
- **$A_{ij} = (-1)^{i+j} |M_{ij}|$ sayısına a_{ij} elemanının eş-çarpanı (kofaktörü) denir.**

Örnek: 3x3 matrisin determinantı,

❶ Bir satır yada sütun seçilir

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

❷ Determinant ifadesi yazılır

$$|A| = a_{11}\ M_{11} + a_{12}\ M_{12} + a_{13}\ M_{13}$$

❸ Kofaktörleri hesaplanır

$$M_{11} = (-1)^{1+1} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}$$

$$M_{12} = (-1)^{1+2} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$$

$$M_{13} = (-1)^{1+3} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

MATLAB'da Matrisin Determinantı

Matrisin determinantını verir.

det (matris)

determinantı hesaplanacak matris

Örnek:

$$A=\begin{bmatrix} -2 & -1 & 4 \\ 6 & -3 & -2 \\ 4 & 1 & 2 \end{bmatrix}$$

```
>> A=[-2 -1 4;6 -3 -2;4 1 2]

A =

     -2     -1      4
      6     -3     -2
      4      1      2

>> det(A)

ans =

    100
```

$$2x + y - 2z = 0$$

$$x - 2y + z = 5$$

$$x + 3y - 2z = -3$$

Lineer denklemi verilmiş olsun. Burada x, y, z değişkenlerini bulmak için önce değişken katsayılarının matrisi ve eşitliğin sağındaki sayılar sütun vektörü biçiminde yazılmalıdır.

Cramer Metodu

```
>>A=[2 1 -2;1 -2 1;1 3 -2]     %değişken katsayıları A matrisi olarak girildi

>> B=[0;5;-3]                  %eşitliğin sağındaki sayılar sütun vektörü olarak girildi

>>m1=A;                        %A matrisi m1 değişkenine atandı
>>m1(:,1)=B                    %m1 matrisinin birinci sütununa B vektörü yazdırıldı
>>m2=A;                        %A matrisi m2 değişkenine atandı
>>m2(:,2)=B                    %m2 matrisinin ikinci sütununa B vektörü yazdırıldı
>>m3=A;                        %A matrisi m3 değişkenine atandı
>>m3(:,3)=B                    %m3 matrisinin üçüncü sütununa B vektörü yazdırıldı
```

```
>>x_y_z=[det(m1);det(m2);det(m3)] / det(A)  %klasik çözüm Cramer Metodu

x_y_z=

    2.2

   -0.4

    2.0
```

Matris Tersi Yöntemi

```
>>x_y_z=inv(A) * B          %matris tersini kullanarak çözüm
x_y_z=
          2.2
         -0.4
          2.0
```

NOT: Sadece kare matrislerin tersleri bulunmaktadır.

Gauss Eliminasyon Yöntemi(\)

>>x_y_z=A \ B %\ operatörü kullanarak gauss eliminasyon tekniği ile çözüm

x_y_z=

2.2

-0.4

2.0

GAUSS-JORDAN ELİMİNASYONU (GAUSS-JORDAN ELIMINATION)

Gauss-Jordan Eliminasyonunda A matrisi Birim matrise dönüştürülecek şekilde işlemler uygulanılır. 3x3 matris için Gauss eliminasyonu uygulanarak aşağıdaki matris elde edilir:

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & b_1^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & b_3^{(2)} \end{bmatrix}$$

Son satırda $a_{33}^{(2)}$ ye bölersek

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & b_1^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{[1]} = 1 & b_3^{[1]} = b_3^{(2)}/a_{33}^{(2)} \end{bmatrix}$$

elde ederiz. Sonrasında 3. satırı $a_{m3}^{(m-1)} (m = 1, 2)$ ile çarparak 1.ve 2. Satırlardan çıkartırsak

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{[1]} = 0 & b_1^{[1]} = b_1^{(0)} - a_{13}^{(0)} b_3^{[1]} \\ 0 & a_{22}^{(1)} & a_{23}^{[1]} = 0 & b_2^{[1]} = b_2^{(1)} - a_{23}^{(1)} b_3^{[1]} \\ 0 & 0 & a_{33}^{[1]} = 1 & b_3^{[1]} \end{bmatrix}$$

sonucunu elde ederiz. Tekrar 2.satırı $a_{22}^{(1)}$ ile bölersek

sonucunu elde ederiz.

2.satırı $a_{22}^{(1)}$ ile bölersek

$$\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & 0 & b_1^{[1]} \\ 0 & a_{22}^{[2]} = 1 & 0 & b_2^{[2]} = b_2^{[1]}/a_{22}^{[1]} \\ 0 & 0 & a_{33}^{[1]} = 1 & b_3^{[1]} \end{bmatrix}$$

sonucunu elde ederiz. 2.satırı $a_{m2}^{(m-1)}(m=1)$ ile çarpıp ilk satırdan çıkartırsak

$$\begin{bmatrix} a_{11}^{(0)} & 0 & 0 & b_1^{[2]} = b_1^{[1]} - a_{12}^{(0)} b_2^{[2]} \\ 0 & 1 & 0 & b_2^{[2]} \\ 0 & 0 & 1 & b_3^{[1]} \end{bmatrix}$$

sonucunu elde ederiz. Burada tekrar ilk satırı $a_{11}^{(0)}$ ile bölersek

$$\begin{bmatrix} 1 & 0 & 0 & b_1^{[3]} = b_1^{[2]}/a_{11}^{(0)} \\ 0 & 1 & 0 & b_2^{[2]} \\ 0 & 0 & 1 & b_3^{[1]} \end{bmatrix}$$

ifadesinde A matrisini birim matrise döndürmüş oluruz. Sağ taraftaki vektörler de çözüm olur.

Editor - C:\Matlab_Dersler\gauss_jordan_elimination.m

gauss_jordan_elimination.m

```
function gauss_jordan_elimination
clc;clear all;warning off;
A=[1 2 5;-1 0 2; 2 1 3];b=[2;0;1];[n m]=size(A);
A=[A b];%genişletilmiş matris

for j=1:n-1 % sütunlar
    for i=j+1:n % satırlar
        if A(i,j)~=0
            lambda=A(i,j)/A(j,j);
            A(i,j:n+1)=A(i,j:n+1)-lambda*A(j,j:n+1);
        end
    end
    A(j,1:n+1)=A(j,1:n+1)/A(j,j);
end
A(n,n:n+1)=A(n,n:n+1)/A(n,n);

for j=n:-1:1
    for i=j-1:-1:1
        lambda=A(i,j);
        A(i,i:n+1)=A(i,i:n+1)-lambda*A(j,i:n+1);
    end
end
x=A(:,n+1);
fprintf('%5s %3s\n','Çözüm','-x-');
fprintf('%12.8f\n',x)
```

VARIABLE

Editor - C:\Matlab_Dersler\gauss_jordan_elim_2.m

gauss_jordan_elimination.m | gauss_jordan_elim_2.m | gaus_gaus_jordan.m | gauss_jordan_elimination.asv

```
function [x,err]=gauss_jordan_elim(A,b)
A = [1 1 1;2 3 5; 4 0 5] % input for augmented matrix A
b = [5 ; 8; 2] % intput for matrix B
[n,m]=size(A); % finding the size of matrix A
err =0; % calculation of error
x=zeros(n,1); % calling fuction zero
if n ~= m
    disp('error: n~=m'); % displaying error if found
    err = 1;
end % end of the scope of if
if length(b) ~= n  % finding the legth of matrix B
    disp('error: wrong size of b'); % displaying error, if found
    err = 2;
else
    if size(b,2) ~= 1
        b=b';
    end % end of the scope of if-else
    if size(b,2) ~= 1
        disp('error: b is a matrix'); % displaying erron in matrix B
        err = 3;
    end
end
if err == 0
    Aa=[A,b];
    for i=1:n
```

Editor - C:\Matlab_Dersler\gauss_jordan_elim_2.m

gauss_jordan_elimination.m | gauss_jordan_elim_2.m | gaus_gaus_jordan.m | gauss_jordan_elimination.asv

```
25 -        for i=1:n
26 -            [Aa(i:n,i:n+1),err]=gauss_pivot(Aa(i:n,i:n+1));
27 -            if err == 0
28 -                Aa(1:n,i:n+1)=gauss_jordan_step(Aa(1:n,i:n+1),i);
29 -            end
30 -        end
31 -        x=Aa(:,n+1);
32 -    end
33 -    A=0;
34      function A1=gauss_jordan_step(A,i) % calling of fuction function
35
36 -    [n,m]=size(A); % determination of size of matrix A
37 -    A1=A; % assigning A to A1
38 -    s=A1(i,1);
39 -    A1(i,:) = A(i,:)/s;
40 -    k=[[1:i-1],[i+1:n]];
41      for j=k
42 -        s=A1(j,1);
43 -        A1(j,:)=A1(j,:)-A1(i,:)*s;
44 -    end % end of for loop
45      function [A1,err]=gauss_pivot(A) % calling of fucntion
46 -    [n,m]=size(A); % finding the size of matrix A
47 -    A1=A; % process of assigning
48 -    err = 0; % error flag
49 -    if A1(1,1) == 0
            check = logical(1); % logical(1) - TRUE
```

Editor - C:\Matlab_Dersler\gauss_jordan_elim_2.m

gauss_jordan_elimination.m | gauss_jordan_elim_2.m | gaus_gaus_jordan.m | gauss_jordan_elimination.asv

```
42 -        s=A1(j,1);
43 -        A1(j,:)=A1(j,:)-A1(i,:)*s;
44 -    end % end of for loop
45      function [A1,err]=gauss_pivot(A) % calling of fucntion
46 -    [n,m]=size(A); % finding the size of matrix A
47 -    A1=A; % process of assigning
48 -    err = 0; % error flag
49 -    if A1(1,1) == 0
50 -        check = logical(1); % logical(1) - TRUE
51 -        i = 1;
52 -        while check
53 -            i = i + 1;
54 -            if i > n
55 -                disp('error: matrix is singular');
56 -                err = 1;
57 -                check = logical(0);
58 -            else
59 -                if A(i,1) ~= 0 & check
60 -                    check = logical(0);
61 -                    b=A1(i,:);          % process to change row 1 to i
62 -                    A1(i,:)=A1(1,:);
63 -                    A1(1,:)=b;
64 -                end
65 -            end
66 -        end
67 -    end
```

Editor - C:\Matlab_Dersler\gaus_gaus_jordan.m

gauss_jordan_elimination.m | gauss_jordan_elim_2.m | gaus_gaus_jordan.m | gauss_jordan_elimination.asv

```
% Code from "Gauss elimination and Gauss Jordan methods using MATLAB"
% https://www.youtube.com/watch?v=kMApKEKisKE

a = [3 4 -2 2 2
     4 9 -3 5 8
     -2 -3 7 6 10
     1 4 6 7 2];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Gauss elimination method [m,n)=size(a);
[m,n]=size(a);
for j=1:m-1
    for z=2:m
        if a(j,j)==0
            t=a(j,:);a(j,:)=a(z,:);
            a(z,:)=t;
        end
    end
    for i=j+1:m
        a(i,:)=a(i,:)-a(j,:)*(a(i,j)/a(j,j));
    end
end
x=zeros(1,m);
for s=m:-1:1
```

Editor - C:\Matlab_Dersler\gaus_gaus_jordan.m

gauss_jordan_elimination.m | gauss_jordan_elim_2.m | gaus_gaus_jordan.m | gauss_jordan_elimination.asv

```
25 -    for s=m:-1:1
26 -        c=0;
27 -        for k=2:m
28 -            c=c+a(s,k)*x(k);
29 -        end
30 -        x(s)=(a(s,n)-c)/a(s,s);
31 -    end
32 -    disp('Gauss elimination method:');
33 -    a
34 -    x'
35      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36      % Gauss-Jordan method
37 -    [m,n]=size(a);
38 -    for j=1:m-1
39 -        for z=2:m
40 -            if a(j,j)==0
41 -                t=a(1,:);a(1,:)=a(z,:);
42 -                a(z,:)=t;
43 -            end
44 -        end
45 -        for i=j+1:m
46 -            a(i,:)=a(i,:)-a(j,:)*(a(i,j)/a(j,j));
47 -        end
48 -    end
```

```
            if a(j,j)==0
                t=a(1,:);a(1,:)=a(z,:);
                a(z,:)=t;
            end
        end
        for i=j+1:m
            a(i,:)=a(i,:)-a(j,:)*(a(i,j)/a(j,j));
        end
    end

    for j=m:-1:2
        for i=j-1:-1:1
            a(i,:)=a(i,:)-a(j,:)*(a(i,j)/a(j,j));
        end
    end

    for s=1:m
        a(s,:)=a(s,:)/a(s,s);
        x(s)=a(s,n);
    end
    disp('Gauss-Jordan method:');
    a
    x'

```

OFFLİNE DERS

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

7.DERS

Arş. Gör. Alev KAYA

Lineer Denklem Sistemlerinin Çözümü

- Sayısal Yöntemler
- A- Jocobi İterasyon Yöntemi
- B- Gauss Seidel Yöntemi
- LAB: Jacobi yöntemi Matlab örnek programı

A- Jocobi(Basit) İterasyon Yöntemi

- **Ardışık(iteratif-dolaylı) yöntemlerde** izlenen yol; denklem sistemlerine yaklaşık bir çözüm takımı ile başlamak ve belirli bir algoritmayı tekrarlayarak, gerçek çözümü **en az hata ile** hesaplamaktır.

- Böylece hem algoritmalarının **kolayca hesaplanabilir** olmaları hem de **yuvarlama hatalarının en az ve iterasyon sayısı arttıkça hata birikimi olmaması** açısından avantajlıdır.

- Bu **yöntemlerin en önemli problemi**; **YAKINSAMA** PROBLEMİDİR.

- Bazı tip problemlerde Jacobi iterasyonu, bazı tip problemlerde ise Gauss-Seidel daha çabuk yakınsar**(GAUSS-SEİDEL, DAHA HIZLI AYNI PROBLEM İÇİNDE).**

Jacobi iterasyonunun yakınsayabilmesi için A matrisinin ana köşegen üzerindeki elemanlarının mutlak değerlerinin bir **koşulu** (strictly diagonally dominant) yerine getirmesi gerekmektedir; A matrisinin i. satırının köşegen üzerindeki değeri (a_{ii})'nin mutlak değeri A matrisinin i. satırındaki tüm elemanların mutlak değerlerinin toplamından büyük olmalıdır.

Aşağıda verilen lineer denklem sistemi Jacobi iterasyonu ile çözülsün;

$$\begin{aligned} 4x_1 - 2x_2 + x_3 &= 6 \\ 4x_1 - 8x_2 + x_3 &= -20 \\ -x_1 + x_2 + 5x_3 &= 11 \end{aligned} \Rightarrow \begin{bmatrix} 4 & -2 & 1 \\ 4 & -8 & 1 \\ -1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -20 \\ 11 \end{bmatrix} \Rightarrow AX = B$$

Yukarıdaki eşitliklerden;

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

elde edilir. Eğer program $x_1^{(0)} = 1;\ x_2^{(0)} = 2;\ x_3^{(0)} = 1$ başlangıç değerleri ile başlar ise;

$$x_1^{(1)} = \frac{6 + 2*2 - 1}{4} = 2.25 \quad ; \qquad x_2^{(1)} = \frac{20 + 4*1 + 1}{8} = 3.125$$

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

elde edilir. Eğer program $x_1^{(0)} = 1;\ x_2^{(0)} = 2;\ x_3^{(0)} = 1$ başlangıç değerleri ile başlar ise;

$$x_1^{(1)} = \frac{6 + 2*2 - 1}{4} = 2.25 \quad ; \qquad x_2^{(1)} = \frac{20 + 4*1 + 1}{8} = 3.125 \quad ;$$

$$x_3^{(1)} = \frac{11 + 1 - 2}{5} = 2$$

birinci iterasyon sonunda bulunan değerler ile ikinci iterasyona gidilir ve bu işlem bu şekilde devam ettirilir ise Tablo 11.1'de gösterildiği gibi 15. iterasyonda (verilen tolerans için);

$$x_1^{(15)} = 3.1745\ ;\ x_2^{(15)} = 4.3333\ ;\ x_3^{(15)} = 1.9683$$

değerleri X çözüm matrisini belirler. İterasyonu durdurmak için kullanılabilecek

bir kriter j. iterasyon sonunda elde edilen $X^{(j)}$ çözüm matrisi ile bundan bir önceki (j-1). iterasyonda elde edilen $X^{(j-1)}$ çözüm matrisi arasındaki farkın mutlak değer olarak başlangıçta belirlenen epsilon değerinden küçük olmasıdır. Buna benzer bir çok durdurma kriteri geliştirilebilir. Ardışık yaklaşımlar için diğer önemli bir nokta da $X^{(0)}$ başlangıç değerlerinin abartılı olarak seçilmesi durumunda yakınsamanın gecikebileceği gerçeğidir. Ardışık yöntemler için verilen sisteme ilişkin uygun değerler (daha önceden) elde edilebilmiş ise bu değerlerin kullanılması yakınsamanın kolaylaşması açısından çok uygun olur.

Tablo 11.1

İterasyon sayısı	x_1	x_2	x_3
1	2.2500	3.1250	2.0000
2	2.5625	3.8750	2.0250
3	2.9312	4.0344	1.9375
4	3.0328	4.2078	1.9794
5	3.1091	4.2638	1.9650
6	3.1407	4.3002	1.9690
7	3.1578	4.3165	1.9681
8	3.1662	4.3249	1.9683
9	3.1704	4.3291	1.9683
10	3.1725	4.3312	1.9683
11	3.1736	4.3323	1.9683
12	3.1741	4.3328	1.9683
13	3.1743	4.3331	1.9683
14	3.1745	4.3332	1.9683
15	3.1745	4.3333	1.9683

$$|4| > |-2| + |1|$$

$$|-8| > |4| + |1|$$

$$|5| > |-1| + |1|$$

JACOBI_BASIT.m

```
% A.X=B lineer matris eşitliğinin JACOBI iterasyonu yöntemi ile çözümü
% A; matrisi N*N boyutunda tekil olmayan katsayılar matrisidir.
% B; matrisi N*1 boyutundadır.
% X; matrisi 1*N boyutunda çözüm matrisinin evriğidir.
% P; matrisi N*1 boyutunda başlangıç değerler matrisidir.
% delta; iterasyonun son iki adımı arasındaki müsade edilebilen fark değeri
% maxiter; max. iterasyon sayısı.Eğer kullanıcı max. sayısına kadar
% iterasyon yakınsamaz ise programı durdurmak için bu değeri kullanılır.

A=[4 -2 1;4 -8 1;-1 1 5];
B=[6;-20;11];
P=[1;2;1];
N=length(B);
X=zeros(1,N);
maxiter=30;
delta=0.00001;
for k=1:maxiter
    for J=1:N
        X(J)=(B(J)-A(J,[1:J-1,J+1:N])*P([1:J-1,J+1:N]))/A(J,J);
    end
    hata=abs(norm(X'-P));
    hatatek=hata/(norm(X)+eps);
    P=X';
    if(hata<delta)|(hatatek<delta)
        'iterasyon maxiterden önce sona erdi'
        break
    end
end
display('iterasyon sayisi=');
display(k-1);
X=X'
```

Command Window

```
>> JACOBI_BASIT

ans =

    'iterasyon maxiterden önce sona erdi'

iterasyon sayisi=
    15


x =

    3.1746
    4.3333
    1.9683

fx >>
```

B- Gauss Seidel Yöntemi

Jacobi iterasyonunda $X^{(0)}$ başlangıç değerleri sırayla;

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

eşitliklerinde yerlerine konulmuştu. Gauss-Seidel yönteminde ise ilk eşitlikte (x_1'e ilişkin) $x_2^{(0)} = 2;\ x_3^{(0)} = 1$ değerleri yerlerine konur. Elde edilen $x_1^{(1)} = 2.25$ değeri (yine aynı iterasyon içindeki)

$$x_2^{(1)} = \frac{20 + 4x_1^{(1)} + x_3^{(0)}}{8} = \frac{20 + 4 * 2.25 + 1}{8} = 3.75$$

eşitliğinde yerine konulur. Yukarıda elde edilen $x_1^{(1)}$ ve $x_2^{(1)}$ değerleri ise $x_3^{(1)}$ eşitliğinde yerine konulursa;

eşitliğinde yerine konulur. Yukarıda elde edilen $x_1^{(1)}$ ve $x_2^{(1)}$ değerleri ise $x_3^{(1)}$ eşitliğinde yerine konulursa;

$$x_3^{(1)} = \frac{11 + x_1^{(1)} - x_2^{(1)}}{5} = 1.9$$

bulunur. Aynı problem Jacobi iterasyonu ile çözüldüğünde ilk iterasyon sonunda;

$$x_1^{(1)} = 2.25 \; ; \; x_2^{(1)} = 3.125 \; ; \; x_3^{(1)} = 2$$

bulunmuştu. Gauss-Seidel yaklaşımında ise

$$x_1^{(1)} = 2.25 \; ; \; x_2^{(1)} = 3.75 \; ; \; x_3^{(1)} = 1.9$$

bulunur. Böyle bir yaklaşım yakınsamayı çabuklaştırarak hesaplamanın daha çabuk bitmesini temin eder. Gauss-Seidel yaklaşımında da Jacobi yaklaşımında olduğu gibi, A matrisinin ana köşegen elemanlarının mutlak değerleri, aynı satır üzerinde yer alan diğer elemanların mutlak değerlerinin toplamından daha büyük olmalıdır. Aksi halde yakınsama sağlanamaz.

Yukarıdaki işlem diğer iterasyonlar içinde yapıldığında elde edilen sonuçlar Tablo 11.2'de gösterilmiştir;

Tablo 11.2

İterasyon sayısı	x_1	x_2	x_3
1	2.2500	3.7500	1.9000
2	2.9000	4.1875	1.9425
3	3.1081	4.2969	1.9622
4	3.1579	4.3242	1.9667
5	3.1704	4.3311	1.9679
6	3.1736	4.3328	1.9682
7	3.1743	4.3332	1.9682
8	3.1745	4.3333	1.9682

Tablo 11.1 ile Tablo 11.2 arasında bir karşılaştırma yapıldığında Gauss-Seidel yaklaşımında çok daha az iterasyon sayısı ile yakınsama sağlandığı görülmektedir. Her iki yaklaşımın aynı probleme aynı ilk koşullar altında uygulandığı da unutulmamalıdır.

Editor - C:\Matlab_Dersler\gauss_seidel.m

JACOBI_BASIT.m | gauss_seidel.m

```
% A.X=B lineer matris eşitliğinin GAUSS_SEİDEL iterasyonu yöntemi ile çözümü
% X; matrisi 1*N boyutunda çözüm matrisidir.
% P; matrisi N*1 boyutunda başlangıç değerler matrisidir.
% delta; iterasyonun son iki adımı arasındaki müsade edilebilen fark değeri
% maxiter; max. iterasyon sayısı.Eğer kullanıcı max. sayısına kadar
% iterasyon yakınsamaz ise programı durdurmak için bu değeri kullanılır.
A=[4 -2 1;4 -8 1;-1 1 5];
B=[6;-20;11];
P=[1;2;1];
N=length(B);
X=zeros(1,N);
maxiter=30;
delta=0.00001;
for k=1:maxiter
    for J=1:N
        if J==1
        X(1)=(B(1)-A(1,2:N)*P(2:N))/A(1,1);
        elseif J==N
            X(N)=(B(N)-A(N,1:N-1)*(X(1:N-1))')/A(N,N);
        else
            X(J)=(B(J)-A(J,1:J-1)*X(1:J-1)-A(J,J+1:N)*P(J+1:N))/A(J,J);
        end
```

```
22 -            end
23 -        end
24 -        hata=abs(norm(X'-P));
25 -        hatatek=hata/(norm(X)+eps);
26 -        P=X';
27 -        if(hata<delta)|(hatatek<delta)
28 -            'iterasyon maxiterden önce sona erdi'
29 -            break
30 -        end
31 -        X;
32 -    end
33 -    display('iterasyon sayisi=');
34 -    display(k-1);
35 -    X=X'
36
```

mmand Window

Command Window

```
>> gauss_seidel

ans =

     'iterasyon maxiterden önce sona erdi'

iterasyon sayisi=
       8


X =

    3.1746
    4.3333
    1.9683

fx >>
```

OFFLİNE

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

8.DERS

Arş. Gör. Alev KAYA

Matris İşlemleri

- Temel Matris işlemleri

- **A-** Matrisin Tersi
- **B-** Matrisin Determinantı
- **C-** Matris Transpozu
- **D-** Matris Normları

- **LAB:** Matrisin Tersini alma Matlab örnek programı

MATLAB/Temel Komutlar

- `clc` Command window'u temizler.
- `clear` İlgili oturumda atanmış tüm değişkenleri siler.
- `clear a` Yalnızca "a" değişkenini siler.
- `demo` Matlab demosunu çalıştırır.
- `date` Gün-Ay-Yıl'ı görüntüler (Örneğin, 17-Oct-2009)
- `exit` Matlab oturumundan çıkar.
- `help` Yardım menüsünü açar.
- `help f_na` f_na fonksiyonu hakkında bilgi verir.
- `save d a` a değişkenini d dosya ismiyle **mat** uzantılı olarak kaydeder.
- `load d` a değişkenini d dosyasından geri çağırır.

Save ve load komutları, matris vb. yapıların kaydedilmesi için çok önemlidir.

MATLAB/Temel dosya türleri

- `*.m` MATLAB program dosyaları
- `*.fig` Grafik dosyaları ve GUI'lerin grafik parçaları
- `*.mat` Değişken ve matris dosyaları
- `*.p` pre-parsed pseudo-code dosyaları (bu dosyaların içeriği görüntülenemez ancak program olarak çağrılabilir, yani MATLAB'de çalıştırılabilir!)

MATLAB/Matrislerin Girilmesi

- **Matris ve vektörler [ ] köşeli parantezleri ile tanımlanır.**
- **Matris ve vektör girmenin 3 farklı yolu vardır:**

Örneğin:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 7 & 8 & 11 \\ 100 & 1 & 4 \end{bmatrix}$$

1.yol

```
A=[1 3 5
7 8 11
100 1 4]
```

2.yol

```
A=[1 3 5;7 8 11;100 1 4]
```

3.yol

```
A(1,1)=1,   A(1,2)=3, A(1,3)=5
A(2,1)=7,   A(2,2)=8, A(2,3)=11
A(3,1)=100, A(3,2)=1, A(3,3)=4
```

MATLAB/Matrislerin Kaydedilmesi

- **Matris ve vektörler *.mat uzantılı olarak `save` komutuyla kaydedilir, `load` ile de istenilen yerden geri çağrılır.**

- **Örneğin, girilmiş bir a matrisini "D:\yildiz" klasörüne "katsayilar.mat" olarak kaydetmek isteyelim: Bunun için aşağıdaki komut dizisi kullanılır;**

```
save D:\yildiz\katsayilar a
```

- **katsayilar.mat olarak kaydedilen a matrisinin herhangi bir zamanda geri çağrılması için,**

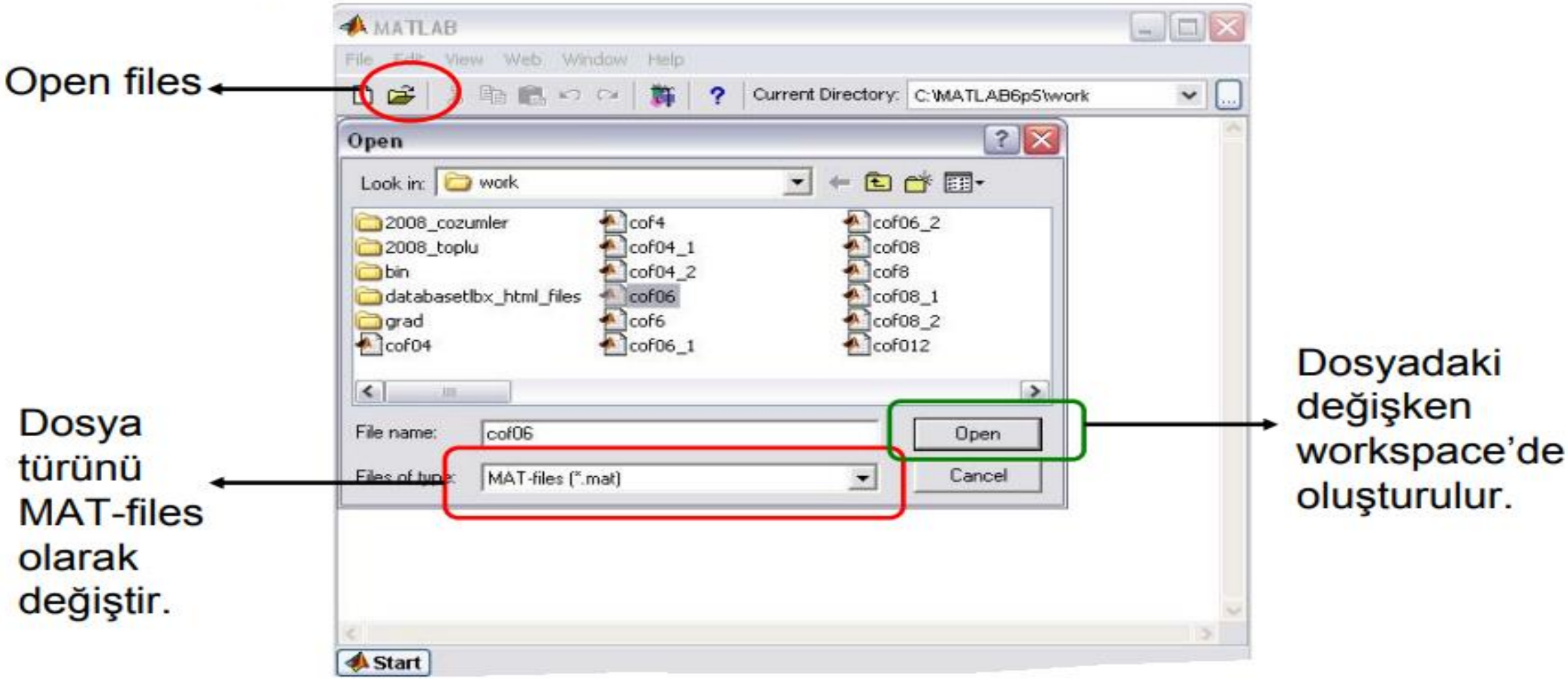
```
load D:\yildiz\katsayilar
```

komut dizisi kullanılır. Geri çağırma işleminden sonra, ilgili matris a dizisi olarak workspace'de kaydedilir (workspace'e kaydetme işleminin geçici olduğunu hatırlayınız!)

Yeni bir matrisi katsayilar.mat olarak kaydettiğimizde, önceki matrisi bir daha görme imkanı kalmaz. Yani save overwrite (üzerine yazma) özelliklidir.

MATLAB/Matrislerin Kaydedilmesi

- ***.mat uzantılı dosyalar, ayrıca MATLAB'den open files kısa yolundan da geri çağrılabilir:**



MATLAB/Sayı Formatları

```
>> format long
>> a=1.12300012312312312;
>> a
a =
```

16 karakter

```
   1.12300012312312
```

```
>> a=100004545.99923423499111;
>> a
a =
```

10^8

```
   1.00004545992342e+008
```

```
>> format short
>> a=1.12300012312312312;
>> a
a =
```

6 karakter

```
   1.1230
```

```
>> a=100004545.99923423499111;
>> a
a =
```

10^8

```
   1.0000e+008
```

Bir sayının istenilen hanesinin gösterilmesi için `fprintf` veya `sprintf` komutları kullanılır:

```
>>a=100004545.99923423499111;
>>fprintf('%1.10f',a)

100004545.9992342300
```

Matris elemanlarının istenilen hanede gösterilmesi için,
printmatrix fonksiyonu oldukça kullanılışlıdır (File-exchange sayfasından download edilebilir!)

MATLAB/Temel lineer cebir komutları

- `inv(a)` Bir a kare matrisinin tersini (inversini) alır.
- `a'` a matrisinin devriğini (transpozesini) alır.
- `det(a)` a matrisinin determinantını hesaplar.
- `a+b` Boyutları aynı olan a ve b matrisini toplar.
- `a-b` Boyutları aynı olan a ve b matrislerinin farkını alır.
- `a*b` Sütun sayısı m olan a matrisiyle satır sayısı m olan b matrisini çarpar.
- `a/b` b düzenli kare bir matrisse (determinantı sıfırdan farklıysa), aynı boyutlu a matrisiyle; `a*inv(b)` işlemini yapar.
- `a.*b` Boyutları aynı olan a ve b matrislerinin elemanlarını karşılıklı olarak çarpar.
- `a./b` Boyutları aynı olan a ve b matrislerinin elemanlarını karşılıklı oranlar.

MATLAB/Temel lineer cebir komutları

- `trace(a)` Bir a matrisinin izini (köşegen elemanlarının toplamını) hesaplar.
- `diag(a)` Bir kare a matrisinin köşegen elemanlarını bir sütun vektöre atar. Ya da a bir vektör ise köşegenleri bu vektörün elemanlarından oluşan bir köşegen matris oluşturur.
- `sum(a)` a matrisinin her bir sütununun toplamını hesaplar. a bir vektör ise sonuç, vektör elemanlarının toplamı olur.
- `triu(a)` Bir matrisin üst üçgen matrisini oluşturur.
- `tril(a)` Bir matrisin alt üçgen matrisini oluşturur.
- `zeros(m,n)` m×n boyutlu sıfır matrisi oluşturur.
- `ones(m,n)` m×n boyutlu elemanları “1” olan matris oluşturur.
- `eye(m)` m×m boyutlu birim matris oluşturur.

MATLAB/Temel matris operatörleri

- `a(:)` a matrisinin sütunlarının ard arda dizilmesinden oluşan bir sütun vektör oluşturur (vec operatörü)
- `a(:,i)` a matrisinin i. sütununu alır.
- `a(j,:)` a matrisinin j. satırını alır.
- `a(:,[i j])` a matrisinin i ve j. sütununu alır.
- `a([i j],:)` a matrisinin i ve j. satırını alır.
- `e=a:b:n` a, (a+b),...,n sayılarından oluşan bir satır vektör oluşturur.

Örneğin,

`e=1:1:n`, 1 ile n arasındaki tam sayılardan oluşan bir vektör.
`e=2:2:n`, 1 ile n arasındaki çift sayılardan oluşan bir vektör.
`e=1:2:n`, 1 ile n arasındaki tek sayılardan oluşan bir vektör.
`e=-10:0.1:n`, -10'dan 0.1 artımla n'ye kadar olan sayılardan oluşan bir vektör.

MATLAB/Temel matris operatörleri

- `length(a)` a matrisinin sütun sayısını verir. a bir vektör ise sonuç, a vektörünün eleman sayısıdır.
- `[m,n]=size(a)` a matrisinin satır sayısını (m) ve sütun sayısını (n) verir.
- `max(a)` Bir a vektörünün en büyük elemanını gösterir.
- `min(a)` Bir a vektörünün en küçük elemanını gösterir.
- `[m,i]=max(a)` Bir a sütun vektörünün en büyük elemanını (m) ve bunun satır numarasını verir.
- `[m,i]=min(a)` Bir a sütun vektörünün en küçük elemanını (m) ve bunun satır numarasını verir.
- `sort(a)` Bir a vektörünün elemanlarını küçükten büyüğe sıralar.
- `a(:,i)=[]` A'nın i. sütununu siler.
- `a(i,:)=[]` A'nın i. satırını siler.

MATLAB/Temel matris operatörleri

- `sortrows(a,i)` Bir a matrisinin elemanlarını i.sütuna göre sıralar.

Örnek:

```
a =
     1     1000
     3       10
     2        5
     4        1
```

```
>> sortrows(a,1)

ans =

     1     1000
     2        5
     3       10
     4        1
```

```
a =
     1     1000
     3       10
     2        5
     4        1
```

```
>> sortrows(a,2)

ans =

     4        1
     2        5
     3       10
     1     1000
```

MATLAB/Uygulama-1

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 7 & 8 & 11 \\ 100 & 1 & 4 \end{bmatrix}$$

Aşağıdaki işlemleri command window'da yapınız.

1) A matrisini giriniz.
2) A matrisinin determinantını hesaplayınız.
3) A matrisinin tersini bulunuz. Çıkan sonucu bir B matrisine atayınız.
4) A*B işlemini yapınız. Elde edilen sonucu irdeleyiniz.
5) A matrisinin 1. sütununu a1, 3. sütununu a3 vektörlerine atayınız.
6) Köşegenleri A matrisinin köşegenlerinden oluşan bir C köşegen matrisi oluşturunuz.
7) a1'in devriği ile a3 vektörünü çarpınız.
8) a1 ile a3 vektör elemanlarını karşılıklı çarpınız.
9) A'nın 3. satırını, diğer satır elemanlarını girmeden, [5 6 7] olarak değiştiriniz.
10) A'nın 1 ve 2. satırlarını siliniz.

MATLAB/Uygulama-1:Çözüm

```
① >> A=[1 3 5;7 8 11;100 1 4];
② >> det(A)
ans =
   -728
③ >> inv(A)
ans =

    -0.0288     0.0096     0.0096
    -1.4725     0.6813    -0.0330
     1.0893    -0.4107     0.0179
>> B=ans;
④ >> A*B
ans =

    1.0000          0     0.0000
         0     1.0000     0.0000
         0     0.0000     1.0000

⑤ >>a1=A(:,1);a3=A(:,3);
⑥ >>C=diag(diag(A));
⑦ >>a1'*a3
ans=
    482
```

Birim matris

```
⑧ >> a1.*a3
ans=
     5
    77
   400

⑨ >>A(3,:)=[5 6 7]
A =

     1     3     5
     7     8    11
     5     6     7

⑩ >> A([1 2],:)=[]
A =

     5     6     7
>>
```

MATLAB/Uygulama-2

$$\mathbf{B} = \begin{bmatrix} 10 & 5 & 5 \\ 70 & 8 & 7 \\ 10 & 1 & 3 \end{bmatrix}$$

Aşağıdaki işlemleri command window'da yapınız.

1) B matrisini giriniz.
2) B matrisini mevcut çalışma klasörünüze **`katsayilar`** ismiyle kaydediniz.
3) Dosyanın kaydedilip kaydedilmediğini kontrol ediniz. (Open Files penceresinden)
4) MATLAB oturumundaki tüm değişkenleri siliniz (clear)
5) Command window'da yazılmış tüm ifadeleri temizleyiniz. (clc)
6) B*2 işlemini yapınız.
7) B matrisini geri çağırınız.
8) B matrisinin üst ve alt üçgen matrislerini oluşturunuz.
9) **`C=[B zeros(3,2)]`** işlemini yapınız.

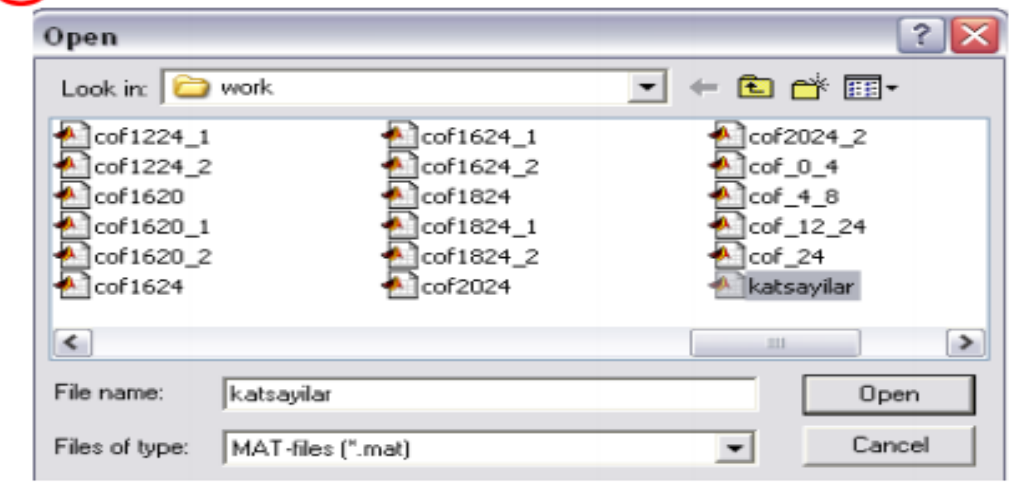
MATLAB/Uygulama-2:Çözüm

```
① >> B=[10 5 5;70 8 7;10 1 3];
② >> save katsayilar B

④ >> clear

⑤ >> clc
```

```
⑥ >> B*2
??? Undefined function or variable 'B'.

⑦ >> load katsayilar
⑧ >> triu(B)
ans =
     10     5     5
      0     8     7
      0     0     3
>>tril(B)
ans =

     10     0     0
     70     8     0
     10     1     3
```

Neden?

③




```
⑨ >> C=[B zeros(3,2)]
C =

     10     5     5     0     0
     70     8     7     0     0
     10     1     3     0     0
```

- `num2str(a)` Bir a sayısını bir karaktere atama (From **num**eric to (**2**) **str**ing)
- `str2num(a)` Karakter olan bir a sayısını sayı değerine atama
- `mat2str(a)` Bir a matrisini bir karakter dizisine atama
- `int2str(a)` Bir a tam sayısını bir karaktere atama
- `char(a)` Bir a hücresini bir karakter dizisine atama
- `cellstr(a)` Bir a karakterini bir hücre dizisine atama
- `num2cell(a)` Bir a sayısını bir hücre dizisine atama

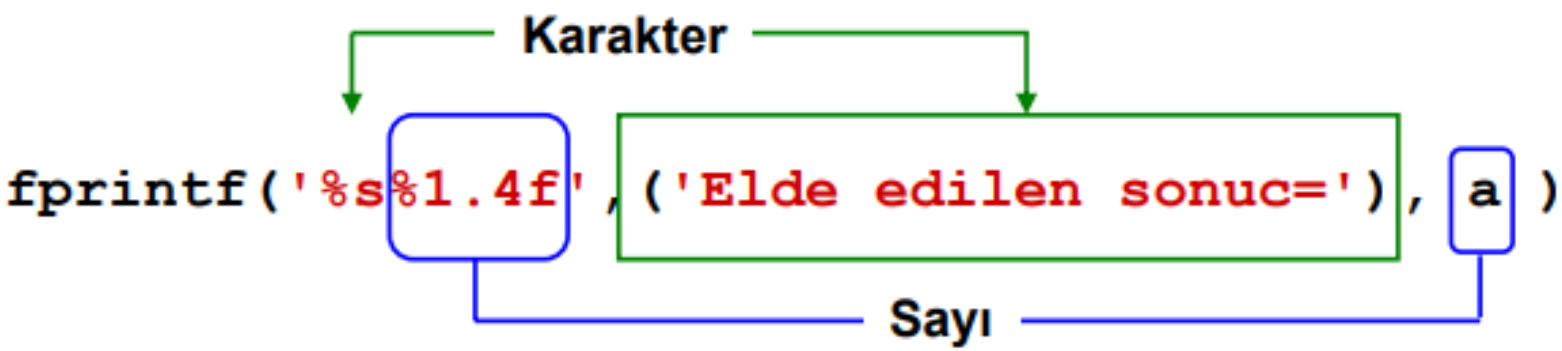
- **Örnek:** Bir işlem sonucunda a=10.234 elde edilsin. "Elde edilen sonuc=10.234" karakterini görüntülemek için,

```
['Elde edilen sonuc=' num2str(a)]
```

Her iki ifade karakter olmalıdır!

yapısı düşünülmelidir.

- Bunun daha gelişmiş biçimi, fprintf ile sağlanır:



MATLAB/Uygulama-3

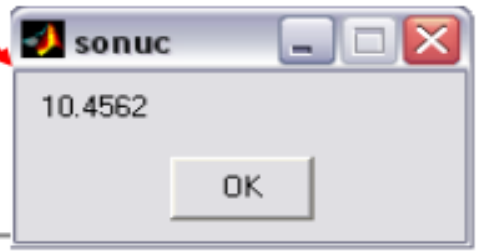
Aşağıdaki işlemleri command window'da yapınız.

1. fprintf fonksiyonunu kullanarak, a=10.45623 sayısını 3 haneye kadar yazdırınız.
2. ['sayinin degeri=' a] ifadesini, a virgülden sonra 2 hane olacak biçimde yazdırınız.
3. Yukarıdaki ifadeyi bir b değişkenine atayınız (sprintf ile)
4. b'nin bir karakter dizisi olup olmadığını denetleyiniz.
5. a değerini önünde 5 karakter boşluk kalacak biçimde 2 haneye kadar yazdırınız.
6. a değişkenini msgbox(a,'sonuc') ifadesiyle bir GUI'ye yazdırınız.
7. b değişkenini msgbox(b,'sonuc') ifadesiyle bir GUI'ye yazdırınız.
8. a'nın karakökünü c değerine atayınız. b ve ['sayinin karakoku', c] ifadesi alt alta olacak biçimde (c, virgülden sonra 5 hane gösterilecek) msgbox içinde yazdırınız.

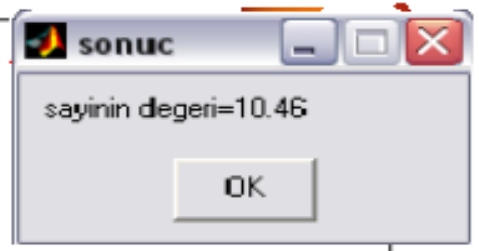
MATLAB/Uygulama-3:Çözüm

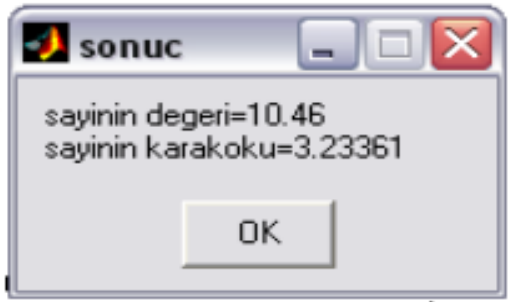
```
① >> a=10.45623;fprintf('%1.3f',a)
  10.456
  >>
② >>fprintf('%s%1.2f',('sayinin...
  degeri='),a)
  sayinin degeri=10.46
③ >>
  >>b=sprintf('%s%1.2f',('sayinin...
  degeri='),a)
④ b =
  sayinin degeri=10.46

⑤ >>ischar(b)
  ans=
       1
⑥ >>fprintf('%10.2f',a)
         10.46
  >>msgbox(num2str(a),'sonuc')
```

5 boşluk+5 karakter




```
⑦ >> msgbox(b,'sonuc')
  >>
⑧ >>c=sqrt(a)
  c =

       3.2336
  >>b1=sprintf('%s%1.5f',('sayinin...
  karakoku='),c);
  >>g=char(b,b1);
  >>msgbox(g,'sonuc')
```





Not: b ve b1 karakter dizilerini alt alta yazdırmanın bir diğer yolu, bunları bir hücre dizisi altında düşünmektir;

```
G=cell(2,1);G{1}=b;G{2}=b1;
msgbox(G,'sonuc')
```

benzer sonucu üretir.

MATLAB/Uygulama-4

Aşağıdaki işlemleri command window'da yapınız.

1. Sonraki işlemlerde kullanılacak bir a sayı değerini, inputdlg fonksiyonu ile girdiren komutu yazınız.
2. a değerinin bir sayı olup olmadığını irdeleyiniz.
3. a*2 işlemini yapınız. Bu işlemin neden sonuç vermediğini irdeleyiniz.
4. a değerini, gerekli ise, sayı dizisine dönüştürünüz.

MATLAB/Uygulama-4:Çözüm

①
```
>> a=inputdlg('Bir sayi giriniz','YTU-2009')
```


```
a =

    '150.123135465'
```

inputdlg ile karakter hücre dizisi oluşturulur. Bu nedenle, girilen verinin sayı yapılması gerekir.

②
```
>>isnumeric(a)
ans =
     0
```

③
```
>>a*2
>>?? Error using ==> *
Function '*' is not defined for values of class 'cell'.
```

④
```
>>a=str2num(char(a))
>>
a =

  150.1231
```

BÖLÜM 4

MATLAB ORTAMINDA VEKTÖR VE MATRİS GÖSTERİMİ

4.1.1. Vektörel sıralama

Öncelikle bir boyutlu sıralamaya örnek teşkil eden satır ve sütun vektörler incelenebilir.

y=sin(x); 0 5≤ x ≤ πT ifadesini tüm x değerleri için hesaplamak imkansızdır, zira 'x' için değer olarak sonsuz sayıda eleman kullanılabilir. Bu nedenle sonlu sayıda 'x' değeri seçilerek;
y = sin(x);
x = 0,0.1π, 0.2π,……,π

eşitliğinde kullanılabilir, y = sin(x) eşitliği İle, çeşitli 'x' değerlerine karşı gelen 'y' değerleri hesaplanabilir. Böyle bir çalışma Tablo 4.1 'de gösterilmiştir.

Tablo 4.1.

X	0	0.lπ	0.2π	0.3π	0.4π	0,5π	0.6π	0.7π	0.8π	0.9π	π
Y	0	.31	.59	.81	.95	1	.95	.81	.59	.31	0

Tablo 4.1'de görüldüğü gibi X1değerine Y1 karşı gelmekte ve bu işlem x(11) değerine y(11) değeri karşı gelinceye kadar devam etmektedir. Alt indis değerleri sıralama ve adresleme açısından bilgi vermektedir. Yukarıdaki ilişki çok açık ve kolay bir şekilde MATLAB ortamına aktarılabilir, Tablo 4.1'de verilen 'x' değerleri satır vektörüne '['-sol köşeli parantez işareti- açılarak tek tek girilir. Her girilen 'x' değeri ile bir sonraki V değeri arasına bir boşluk bırakılabileceği gibi virgül de kullanılabilir. 'x' girişleri sona erdiğinde ']'sağ köşeli parantez işareti kullanılarak vektör tanıtımı tamamlanır.

```
>> y=sin(x);
>> x=[0 0.01*pi .2*pi .3*pi .4*pi .5*pi .7*pi .8*pi .9*pi pi];
>> x(3)
ans =
 0.6283
>> y(3)
ans =
  0.5878
```

şeklinde matlab ortamında gösterilir.

4.1.2.Kolon oparatörü(:) kullanarak vektör elde edilmesi

Kolon operatörü yardımı ile vektör elde *etmek* için kullanılan format türleri aşağıda gösterilmiştir.

-(başlangıç değeri:artış değeri:son değer)

```
>> A=5:0.5:8

A = 5.0000   5.5000   6.0000   6.5000   7.0000   7.5000   8.0000
```

-[başlangıç değeri: artış değeri: son değer]

```
>> A=[5:0.5:8]

A =
```

5.0000 5.5000 6.0000 6.5000 7.0000 7.5000 8.0000

>> A=[5:0.5:8]' satır vektörü sütun vektörüne dönüşür.

A =

5.0000
5.5000
6.0000
6.5000
7.0000
7.5000
8.0000

4.1.3.Mevcut bir vektörün elemanları kullanılarak başka bir vektör elde edilmesi

C vektörü, daha önce tanımlanmış olan y vektör elemanları içinden bazıları seçilerek oluşturulabilir.
y = 0 0.0314 0.5878 0.8090 0.9511 1.0000 0.8090 0.5878 0.3090 0.0000

>> C=y(2:1:4) ikinci elemandan bir artış ile dördüncü elemana kadar vektör oluşturma

C = 0.0314 0.5878 0.8090

>> H=x(7:end) yedinci elemandan başlayıp sona kadar gider.

H =2.1991 2.5133 2.8274 3.1416

>> K=y([7 6 2 1]) parantez içinde rakamlar ise sadece vektörden o değerler alınır.

K =0.8090 1.0000 0.0314 0

4.1.4. Vektör oluşturmanın diğer yöntemleri

Birleştirme: MATLAB ortamında daha önce tanımlanmış bir vektörü kullanarak yeni bir vektör oluşturulabilir. Aşağıda verilen örnek İncelenmelidir.

» A=[1.5, 2.6, 8.9];

» B=*[4.2* A]

B =4.2000 1.5000 2.6000 8.9000

Yukarıda verilen örnekte görüldüğü gibi daha önce tanımlanan A vektörünü kullanarak B vektörü oluşturulmaktadır. B vektörü *4.2* değerli elemanı birinci eleman olarak almakta ve geri kalan elemanları ise (A vektörünün sıralamasını aynen koruyarak) A vektöründen almaktadır.

Değiştirme: MATLAB ortamında daha önce tanımlanmış bir vektörün bazı elemanlarım yeniden belirlemek mümkündür. Aşağıda verilen örnekte bu görülebilir.

»B*(4)*= -3,5

»B

B=4.2000 1.5000 2.6000 -3.5000

Yukarıda verilen örnekte görüldüğü gibi daha Önce tanımlanan *B* vektörünün *4* numaralı elemanına yeni bir değer *(-3.5)* atanmakta ve *B* vektörü daha önceki değerinden farklı bir değer almaktadır.

Genişletme: MATLAB ortamında daha önce tanımlanmış bir vektöre bazı yeni elemanların ilave edilmesi mümkündür. Aşağıda verilen örnek incelenmelidir.

» B(5}= 6.9;

»B

B=4.2000 1.5000 2.6000 -3.5000 6.9000

Yukarıda verilen örnekte görüldüğü gibi daha önce tanımlanan *B* vektörüne 6.9 değerinde *5.* eleman ilavesi yapılmakta ve *B* vektörü daha önceki değerinden farklı bir değer almaktadır.

Bir vektörün bir sayı ile işleme sokulması sonunda yeni bir vektör elde edilmesi

*>> E=(0:0.4:2)*pi*

E =0 1.2566 2.5133 3.7699 5.0265 6.2832

>> N=(0:0.4:2)+pi

N = 3.1416 3.5416 3.9416 4.3416 4.7416 5.1416

linspace(başlangıç değeri,son değer,toplam sayı)

>> F=linspace(0,pi,13) 0 ile pi arası eşit artış ile 13 sayının oluşması

F =Columns 1 through 12

0 0.2618 0.5236 0.7854 1.0472 1.3090 1.5708 1.8326 2.0944 2.3562 2.6180 2.8798

Column 13

3.1416

logspace *(başlangıç değeri,son değer,toplam sayı)* komutu kullanarak vektör üretilmesi:

Yukarıda verilen komut ile oluşturulan vektör matris elemanları üstel (10 üstü) sayılardan oluşmaktadır.

>> A=logspace(0,2,11)

A =1.0000 1.5849 2.5119 3.9811 6.3096 10.0000 15.8489 25.1189 39.8107

63.0957 100.0000

4.1.5.Vektör uzunluğu

MATLAB ortamında verilen bir vektörün *eleman sayısını* bulmak için length komutu kullanılır

>> length(y)

ans =10

4.1.6.Sütun vektör oluşturulması

Şimdiye kadar gösterilen vektörler satır vektörü türündeydi. Satır vektörü (m elemanlı) m*1 boyutu ile simgelenir. Sütun vektörü ise (m elemanlı) l*m boyutu ile gösterilir. Aşağıda verilen örnekte görüldüğü gibi bir satır vektörünün elemanları arasına (;) işareti konularak sütun vektörü elde edilebilir.

>> a=[1;3;7;8;9]

a = 1

3

7

8

9

Yukarıda verilen a vektörünün bir elemanı ekrana yazıldıktan sonra ('enter') tuşu kullanılarak da sütun vektör oluşturulabilir.>> a=[1

2

3

4

5]

Satır vektöründen sütun vektörü elde edilmesinin bir diğer yolu ise transpoze işlemidir. MATLAB ortamında *transpoze* işlemi (') işareti ile gerçekleştirilir.

```
>> a=[1 2 3 4 5];
>> b=a';
>> b
b= 1
    2
    3
    4
    5
>> f=linspace(0,pi,13)'
f =    0
   0.2618
   0.5236
   0.7854
   1.0472
   1.3090
   1.5708
   1.8326
   2.0944
   2.3562
   2.6180
   2.8798
   3.1416
```

Eğer *kompleks* bir *vektöre* (z) transpoz işareti (') uygulanırsa elde edilecek yeni vektör (y=z')> z vektörünün *eşlenik transpozu* olacaktır.

```
>> z=[1+3j 4-2j -2-5j];
>> y=z';
>> y
y =1.0000 - 3.0000i
   4.0000 + 2.0000i
  -2.0000 + 5.0000i
```

4.1.7. Vektörün 0 veya 1 sayılarından oluşması

zeros(m,l): m adet elemanının tümü sıfır (0) olansütun vektörü.

```
>> x=zeros(3,1)
x =0
    0
    0
```

ones(m,l):m adet elemanın tümü bir (l) olan sütun vektörü:

```
>> y=ones(1,3)
y =     1     1     1
```

4.2. Matris oluşturulması

Bir matris iki boyutlu bir elemandır, satır ve sütun sayılan birden çok olabilir. MATLAB ortamında bir matris oluşturmak için dört adım yeterlidir:

1. Sol köşeli parantez işareti' [' aç.
2. Satır elemanlanın ya aralarında boşluk bırakarak yada virgül kullanarak köşeli parantez içine yaz.

Satır bitiminde ya noktalı virgül';' ya da ('enter') tuşu kullanarak diğer satıra geç.

En son eleman yazıldıktan sonra sağ köşeli parantez ']' kullanarak matrise eleman girişini sona erdir.

```
>> g=[1 2 3;4 5 6]
g =1     2     3
    4     5     6

>> x=1:4;
>> y=4:4:16;
>> L=[x'  y']
L = 1     4
    2     8
    3    12
    4    16
```

4.2.1. Matris elemanlarının adresleri

>> d=[1 2 3;4 5 6];

>> d(2,3) satır ve sütun numarası yazılarak istenen değere ulaşılmıştır.

ans =

6

*4.2.2. Matris elemanlarının **MATLAB** ortamında saklanması*

MATLAB ortamında kullanılan bir matris MATLAB arka planında bir dizi olarak saklanır.

>> h=[1 2 3 4;5 6 7 8;9 -8 -7 -6]

h = 1 2 3 4

5 6 7 8

9 -8 -7 -6

Bu arka planda h=(1 5 9 2 6 -8 3 7 -7 4 8 -6)

4.2.3.Matris elemanlarının bir kısmı ile başka bir matris oluşturulması

MATLAB ortamında daha önce tanımlanmış bir matrisin bazı satır veya sütunlarını kullanarak yeni matrisler elde etmek mümkündür.

>> a=[1 2 3

4 5 6

7 8 9]

a = 1 2 3

4 5 6

7 8 9

> b=a(:,2) virgül solda olduğu için sütun işlemi sağda olsa satır işlemi olur.

b = 2

5

8

>> c=a(1,:)

c =

1 2 3

>> d=a(1:3, :) virgülün solunda yer alan *1:3* işareti, *a* matrisinde *1.* satırdan *3.* satıra kadar tüm satırların *d* matrisine atanacağını göstermektedir

d = 1 2 3

4 5 6

7 8 9

>>

>> k=h(2:3,1:2) h matrisinde 2. ve 3. Satır ile 1. ve 2. Sütun ların kesişim elemanları k matrisine atanır.

k = 5 6

9 -8

4.2.4. Matrisleri birleştirerek yeni bir matris oluşturulması

MATLAB ortamında mevcut matrisleri kullanarak yeni bir matris oluşturulması işlemi yapılırken dikkat edilmesi gereken en önemli nokta birleştirilen matrisler arasındaki boyut uyumudur. Burada matris oluşturmak için rand komutundan faydalanılacaktır

>> a1=rand(3,2)

a1 =0.8147 0.9134

0.9058 0.6324

0.1270 0.0975

>> a2=rand(3,4)

a2 =0.2785 0.9649 0.9572 0.1419

0.5469 0.1576 0.4854 0.4218

0.9575 0.9706 0.8003 0.9157

>> a1_2=[a1 a2]

a1_2 =0.2785 0.9649 0.9572 0.1419 0.7922 0.0357 0.6787 0.3922

0.5469 0.1576 0.4854 0.4218 0.9595 0.8491 0.7577 0.6555

0.9575 0.9706 0.8003 0.9157 0.6557 0.9340 0.7431 0.1712

4.2.5.Matris büyüklükleri

size (A) : A matrisinin satır ve sütun sayısı hakkında bilgi verir. İlk sayı satır sayısını, ikinci sayı ise sütun sayısını verir.

[r,cl=size(A):A. matrisinin satır sayısı r, sütun sayısını ise c adlı değişkene atanır.

r=size(A,1) : A matrisinin satır sayısı r adlı değişkene atanır ve sütun sayısı ile ilgilenilmez.

c=size (A, 2) :A matrisinin sütun sayısı c adlı değişkene atanır, satır sayısı ile ilgilenilmez.

n=length(A) : A matrisinde satır veya sütün sayısından hangisi daha büyük ise bu sayıyı n adlı değişkene atanır.

4.2.6.Matriste 0 ve 1 işlemleri

L=zeros(n): n*n boyutunda sıfir elemanlarından oluşan bir L matrisi inşa eder

T=zeros(m,n): m*n boyutunda sıfir elemanlarından oluşan bir T matrisi oluşturur.

T=zeros(size(A)):A matrisi ite aynı boyutta fakat tüm elemanları sıfır olan bir T matrisi yapar.

L=ones(n): n*n boyutunda bir elemanlarından oluşan bir L matrisi inşa eder.

T=ones(m,n): m*n boyutunda bir elemanlarından oluşan bir T matrisi yapar.

T=ones(size(A)): A matrisi ile aynı boyutta fakat tüm elemanları 1 olan bir T matrisi yapar.

T=eye(m,n) :Köşegen eleman değerleri 3, diğer tüm elemanları 0 olan m*n boyutunda olan

4.2.7. MATLAB ortamında tanımlı bir matrisin yeniden düzenlenmesi

Tüm matrisler MATLAB arka planında bir boyutlu bir dizi olarak saklanır. Kullanıcı MATLAB ortamında verilen bir matrisin boyunu (eleman değerleri aynı kalmak şartı ile) değiştirmek istediğinde yukarıda bahsedilen gerçeği unutmamalıdır.

>> N=[1 2 3 4;5 6 7 8;9 10 11 12]

N =1 2 3 4

5 6 7 8

9 10 11 12

>> N(:) Bu matrisi matlab arka planda aşağıdaki şekilde görülür.

ans =1

5

9

2

6

10

3

7

11

4

8

12

M=reshape (N,a,b):N matrisinin elemanlarını a satır sayılı b sütun sayılı M adlı matris içinde saklar.

>> M=reshape(N,4,3)

M =1 6 11

5 10 4

9 3 8

2 7 12

K=f liplr (M): M matrisinin satır elemanlarını 180 derece ters çevirerek K adlı matris içine atar.

>> K=fliplr(M)

K =11 6 1

4 10 5

8 3 9

12 7 2

T=flipud (K) :M matrisinin satır sıralamasını ters çevirerek T adlı matris içine atar

```
>> T=flipud(K)

T =  12     7     2

8     3     9

4    10     5

11     6     1
```

S=rot90 (T): T matrisini 90 derece saat ibresinin hareket yönünü ters yönünde döndürür

ve elde edilen matrisi S adlı matrise atar.

```
>> S=rot90(T)

S = 2     9     5     1

    7     3    10     6

   12     8     4    11
```

4.2.8.Matris ve sayıların birlikte işleme girmesi

Bir matrisin bir *sayı* ile toplamı, farla, çarpımı veya bölümü demek, o *sayı* ile tüm matris elemanları arasında bu işlemlerin yapılması demektir.

```
N =1     2     3     4

    5     6     7     8

    9    10    11    12

>> g=2*N


g = 2     4     6     8

   10    12    14    16

   18    20    22    24
```

4.2.9. İki vektör elemanlarının birbirleri ile işleme girmesi

İki vektörün boyutu aynı olduğunda iki vektörün elemanları arasında (bire-bir) dört işlem yapılabilir. Eğer boyutlar farklı ise hata mesajı ile karşılaşılır.

Toplama a+b

Çıkarma a-b

*Çarpma a.*b*

Bölme a./b

Üstel a.^b

```
>> A=[1 2 3];

>> B=[2 4 6];

>> K=A-B

K =-1    -2    -3
```

*>> M=A.*B*

M =2 8 18

>> C=A.^B

C =1 16 729

4.2.10.Çok boyutlu matris yapıları

Buraya kadar kullanılan matrisler iki boyuta sahipti. MATLAB ortamında çok boyutlu matris yapıları da tanımlıdır. Üç boyutlu bir matrisi tanımlamak için şöyle bir örnek verilebilir: Üç boyutlu matrisin ilk iki boyutu kitabın bir sayfasını (x,y-düzlemi), üçüncü boyutu ise bu sayfadan sonraki (arkaya doğru) herhangi bir sayfayı temsil etsin (z ekseni).Örnek olarak bl matrisi 2 satır 3 sütun ve 2 sayfadan oluşan bir eser olsun, bl matrisinin ilk sayfası;

>> b1(:,:,1)=[1 3 5;7 9 11] 1. Boyut ilk sayfa

b1 =1 3 5

7 9 11

>> b1(:,:,2)=[0 2 4;6 8 10] 2. Boyut ikinci sayfa

b1(:,:,1) = 1 3 5

7 9 11

b1(:,:,2) =0 2 4

6 8 10

>> ndims(b1) b1 matrisinin kaç boyutlu olduğu sorulur

ans =3

>> size(b1) b1 matrisi ölçüsü sorgulanır.

*ans =2 3 2 2*3*2 ölçülerinde olduğu anlaşılır.*

4.3.Logaritmik eksen takımlarında çizim

plot(x,y): x ve y eksenindeki değişkenlerin eksenler üzerinde lineer olarak dağıldığı kabulü ile yatay eksendeki (x) değerler ile düşey eksendeki (y) değerler arasında gerçekleşen değişimi çizer.

semilogx(x,y): x ekseni üzerindeki değişken değerlerinin bu eksen üzerinde logaritmik olarak yayıldığı, y ekseni üzerindeki değişken değerlerinin ise bu eksen üzerinde doğrusal olarak yayıldığı kabulü ile x ve y değerleri arasındaki değişimi çizer

semilogy(x,y): y ekseni üzerindeki değişken değerlerinin bu eksen üzerinde logaritmik olarak yayıldığı, x ekseni üzerindeki değişken değerlerinin ise bu eksen üzerinde doğrusal olarak yayıldığı kabulü ile x ve y değerleri arasındaki değişimi çizer.

loglog(x,y): Hem x ekseni hem de y ekseni üzerindeki değerlerin logaritmik olarak bu eksenler üzerine yayıldığı kabulü ile x ve y değerleri arasındaki değişimi çizer.

Aynı eksen takımı üzerine çizilen eğrileri birbirinden ayırmak için kullanılabilecek diğer bir yaklaşım ise eğrilerin yapısının farklı kılınmasıdır. Örneğin eğri, kesik olmayan bir çizgiden meydana gelebilir, noktalı çizgilerden meydana gelebilir, bir nokta-bir çizgiden meydana gelebilir veya iki çizgi yan yana getirilerek eğri oluşturulabilir. Aşağıda (daha önce değerleri hesaplanmış olan) t ve h değişkenleri arasındaki değişimin çizildiği farklı eğri çizim türlerini gösteren MATLAB komutları tanıtılmıştır:

»plot (t,h, ':') -t ile h arasındaki değişimi gösteren eğri '….. ' şeklinde çizilir-

»plot(t,h,'-') -tile h arasındaki değişimi gösteren eğri '___ ' şeklinde çizilir-

»plot (t,h,' -.') -t ileh arasındaki değişimi gösteren eğri-.-.-.-. şeklinde çizilir-

»plot(t,h,' - -') -t ile h arasındaki değişimi gösteren eğri '---- ' şekline çizilir.

Problem 4.1

PROBLEM:Bir cisim V ilk hızı ile yukan doğru fırlatılmaktadır. Yer çekimi 'g' ile gösterilmektedir. Havanın cisme gösterdiği direnç ihmal edilmektedir. Cismin atıldığı andan düştüğü ana kadar olan zaman süresince, cismin yerden yüksekliğinin (h) zamana (t) göre değişimini çiziniz.

Çözüm

Cismin yerden yüksekliğinin uçuş zamanına göre değişimi;

$h(t)=vt-0.5gt^2$

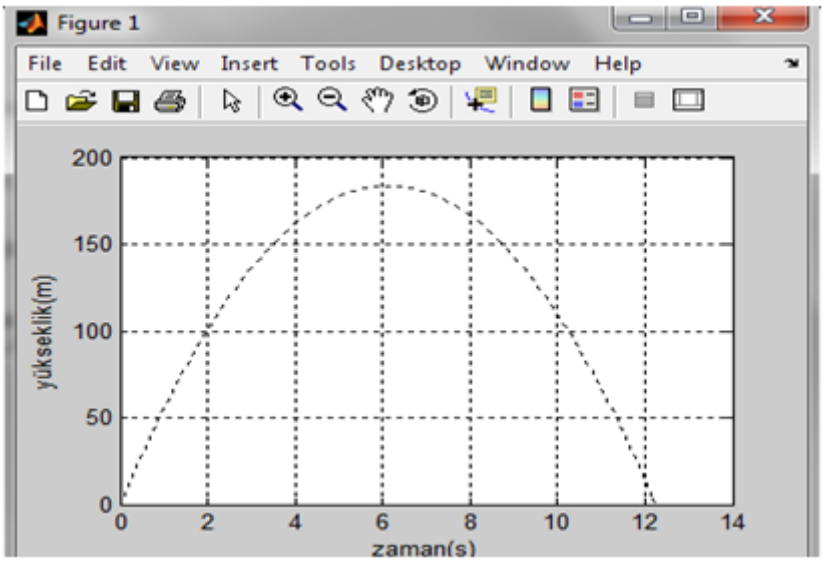
olur. Cismin uçuş süresi tu ile gösterilirse, cisim düştüğünde h=0 olacağından; $h(tu) = vtu-0.5gt^2=0$

yazılabilir. Yukarıda verilen eşitlikten uçuş süresi;

$t_u=2v/g$

olur.Eger v = 60m/s, g = 9.8m/sn^2 alınırsa çizim için gereken MATLAB programı aşağıda verildiği gibi olmalıdır. (program MATLAB editör ortamında yazılmıştır):

```
>> g=9.8;
>> v=60;
>> tu=2*v/g;
>> t=linspace(0,tu,256);
>> h=v*t-g/2*t.^2;
>> plot(t,h,'k:');
>> xlabel('zaman(s)')
>> ylabel('yükseklik(m)')
>> grid
```



Yukarıda verilen ve MATLAB editöründe yazılan programda `plot(t,h,'k:')` ifadesinde yer alan k değeri çizimin siyah renkli olacağını, (:) işareti ise çizimin noktalı olarak yapılacağını göstermektedir, `grid` komutu ise çizilen şeklin arka planım 'ızgara' haline getirmektedir.

4.4.Aynı eksen takımım üzerinde birden çok eğrinin çizilmesi(Yatay eksenin(x) ortak,düşey eksenin(y)farklı değerler alması)

plot(x,y,x,z,x,w,…): Bu tür bir çizim komutunda 'yatay' eksende ortak fakat 'düşey' eksende farklı değerler alan y = f1 (x), z = ***f2*** (x), w = f3 (x) gibi eğriler çizilebilir. Bu çizim komutu içinde yer alan x, y, z, w gibi değişkenler birer vektör'dür.

plot(x,F):Bu tür bir çizim komutunda x vektör matrisi, F matrisini oluşturan sütun matrislerden biri veya birkaçıdır. 'Yatay' eksen ortaktır. x'in boyutu F'in satır sayısı ile aynı ormandır. Bu komut da, plot (x,y,x,z,x,w….) komutuna benzer: y; F matrisinin bir sütunu, z; F matrisinin diğer bir sütunu, w; ise F matrisinin bir başka sütunu gibi düşünülebilir.

plotyy(x,f1,x,f2): Bu tür bir çizim komutunda x vektör matrisi, F matrisini oluşturan sütun matrislerden biri veya birkaçıdır. 'Yatay' y eksenleri olarak çizim penceresinin sol ve sağ tarafları alınır. Böylece en çok iki tane f fonksiyonu çizilebilir, plotyy komutandaki iki adet y harfi de bu durumu göstermektedir. Bu tür çizim komutunda her iki eğri farklı renkte çizilir, f 1 eğrisi çizim penceresinin sol tarafına, f 2 eğrisi ise eğri penceresinin sağ tarafına yerleştirilir.

legend('y','z','w',A): Bu komut çizimin yapıldığı pencere İçinde bir kutu açar. Bu kutunun İçine y,z,w... gibi düşey eksen üzerine çizilen eğrilerin isimleri ve bu eğrilerin çizim türü (renk ve çizgi biçimi) hakkında bilgi yazılır. Böylece okuyucunun farklı eğrileri tanıması mümkün olur. Bu kutunun çizilen şekil ekranı üzerinde yerleştirileceği yer ise A hanesine yazılan rakamın aldığı değere bağlı olarak değişir. Tablo 4.3'te kutunun, A'nın aldığı değerlere göre şekil penceresinde değişen konumu gösterilmiştir.

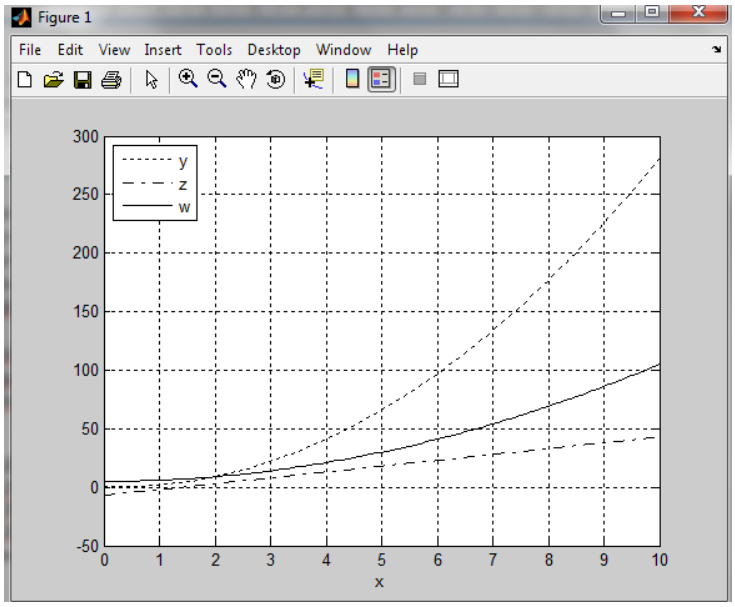
y = 3x2-2x+l

z=5x-7

w = x2 +5

eğrilerini *x* [0;10] aralığında matlab ta çizelim.

```
>> x=0:0.1:10;
>> y=3*x.^2-2*x+1;
>> z=5*x-7;
>> w=x.^2+5;
>> plot(x,y,'k:',x,z,'k-.',x,w,'k-');
>>
>> xlabel('x')
>> grid
>> legend('y','z','w',2)
```



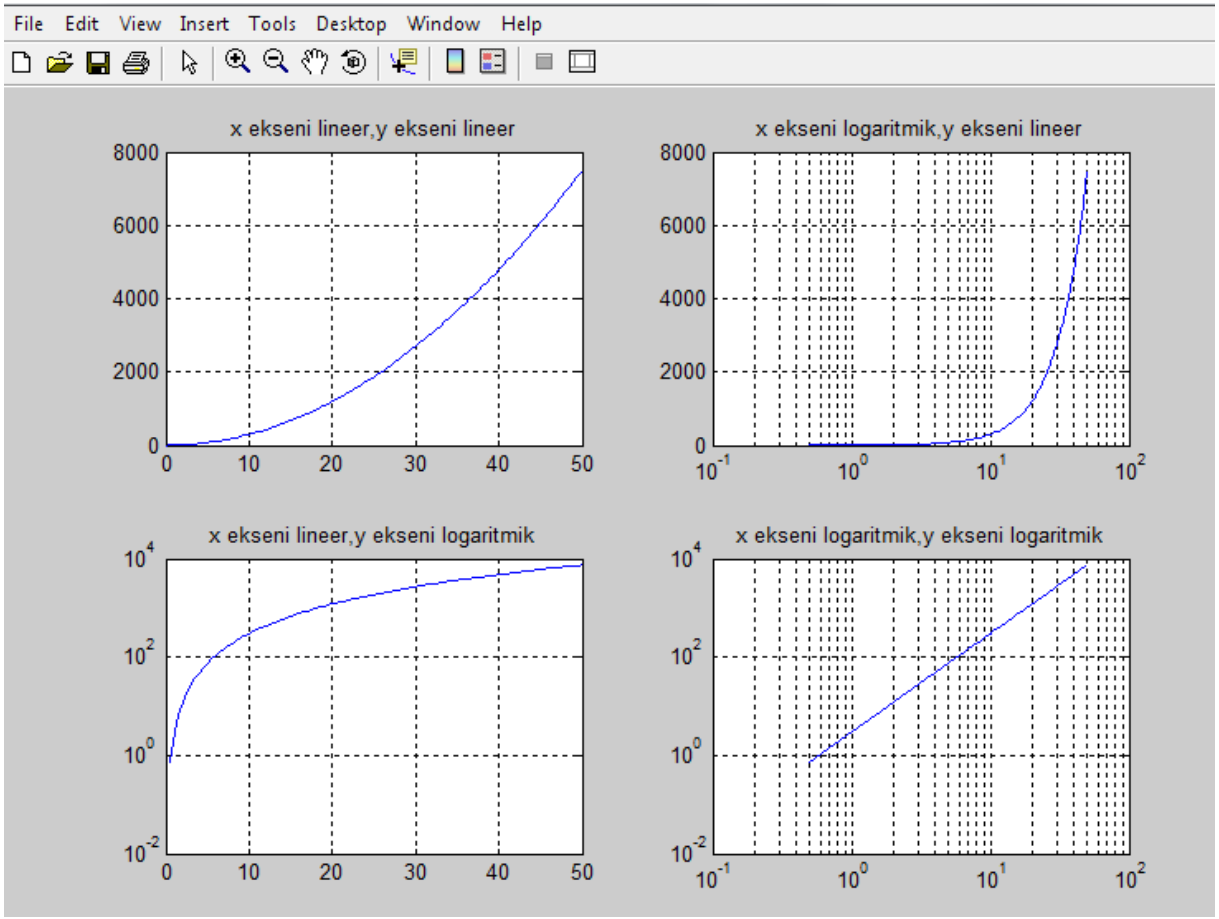
4.5.Ekranın birden çok çizim için pencerelere ayrılması

subplot komutu ekranı birden çok alt çizim ekranlarına ayırır. Örnek olarak; eğer iki tane çizim aynı ekran üzerinde yapılacak ise bunlar ya (yan yana) üst sağ ve üst sola yerleştirilir yada (üst üste) ait sağ ve alt sola yerleştirilir. Eğer dört adet çizim yapılacak ise İki adet üste, iki adet alta çizim yapılabildiği gibi 4 tanesi yan yana da çizdirilebilir. Bu komut; subplot(n,m,p) biçiminde kullanılır. Burada çizim yapılacak şekilleri bir matrisin elemanları olarak görmek mülkündür. Bu durumda n; satır sayısını, m; sütun sayısını gösterir, p ise çizilen alt pencerenin kaçıncı alt pencere olduğunu gösterir. Pencere sayısı soldan sağa ve yukandan aşağıya doğru numaralanır. Örnek olarak, subplot (2,1,1) komutu yazıldığında bu komutu takip eden plot komutu ile çizdirilecek olan şekil, (ekran bir matris olarak düşünülerek) matrisin (n=2) ikinci satır (m=1) birinci sütununa yerleştirilir. Bu şekil (p=l) ilk şekil olacaktır. Örnek olarak subplot (222) yazıldığında bu komutu takip eden plot komutu İle verilen şekil dört parçaya ayrılan ekranın (İkinci satır İkinci sütun) sağ altına yerleştirilir ve bu ikinci şekil (p=2) olmalıdır.

$y=3x^2-1$

fonksiyonunu plot (x,y), semilogx (x,y), semilogy (x,y) ve loglog(x,y) komuttan yardımı ile 0:50 aralığında farklı eksen takımlarında çizelim.

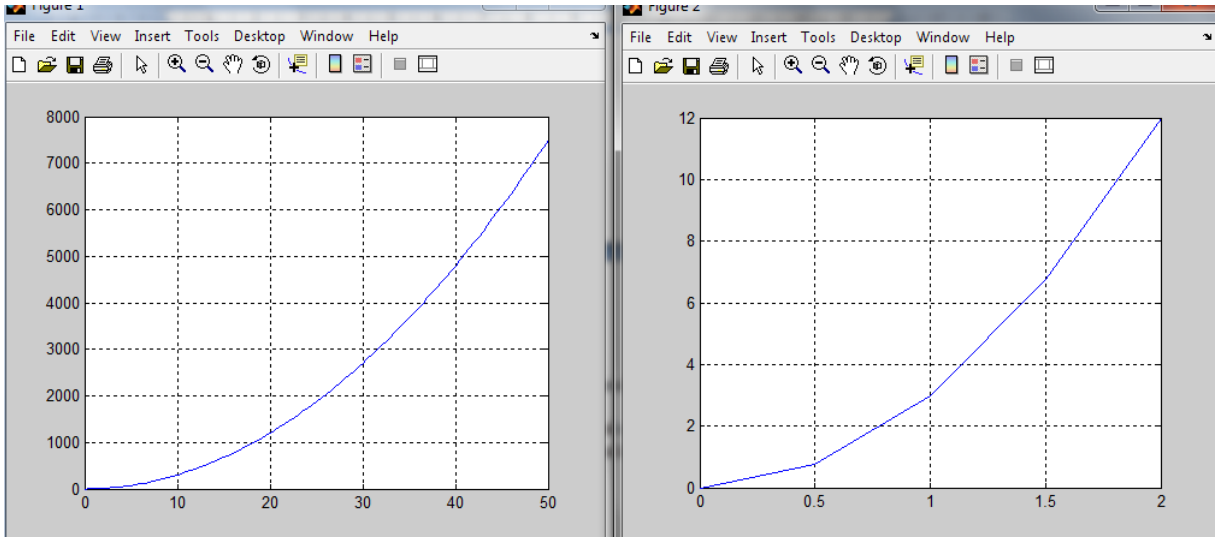
```
>> x=0:0.5:50;
>> y=3*x.^2;
>> subplot(2,2,1),plot(x,y);
>> title('x ekseni lineer,y ekseni lineer'),grid;
>> subplot(2,2,2),semilogx(x,y);
>> title('x ekseni logaritmik,y ekseni lineer'),grid;
>> subplot(2,2,3),semilogy(x,y);
>> title('x ekseni lineer,y ekseni logaritmik'),grid;
>> subplot(2,2,4),loglog(x,y);
>> title('x ekseni logaritmik,y ekseni logaritmik'),grid;
```



4.6.Plot komutu kullanılarak eğrinin daha dar aralıkla çizdirilmesi

Kullanıcı plot komutu ile çizdirmek istediği y=f(x) fonksiyonunda (program satırları içinde x'i daha genişaralıkta belirlese bile) x'i daha dar aralıkta da çizdirebilir.

```
>> figure(1)
>> x=0:0.5:50;
>> y=3*x.^2;
>> plot(x,y),grid;
>> figure(2)
>> plot(x(1:5),y(1:5)),grid;
```



BÖLÜM 5

MATEMATİKSEL FONKSİYONLAR

5.1. Periyodik Fonksiyonlar

Periyodik fonksiyon;

$x(t)= (t + nT)$, $n=1,2,3,\ldots$ (5.1)

koşulunu sağlar. (5.1) ifadesinde T; periyot olarak adlandırılan sabit ve pozitif bir değerdir. Her T süre sonunda sinüzoidal işaret aynen tekrarlanır. Eğer periyodik olan fonksiyon sinüzoidal olan;

$x(t)= A \cos (wt +\theta)$ (5.2)

fonksiyonu ile değişiyor ise bu ifade kullanılan değişkenler aşağıda gösterilmiştir:

A: Değişkenin genliğidir. Pozitif ve negatif alternansların maksimum değerleri arasındaki fark 2A olur. Birimi x'in gösterdiği büyüklüğe göre değişir. Eğer x gerilimi gösteriyor ise A'nın birimi volt olur.

t : Bağımsız zaman değişkenidir ve birimi saniyedir (s).

w: Açısal frekanstır, birimi radyan/saniye (rad/s) olup, periyot ile arasında $T = 2\pi / w$ ilişkisi vardır.

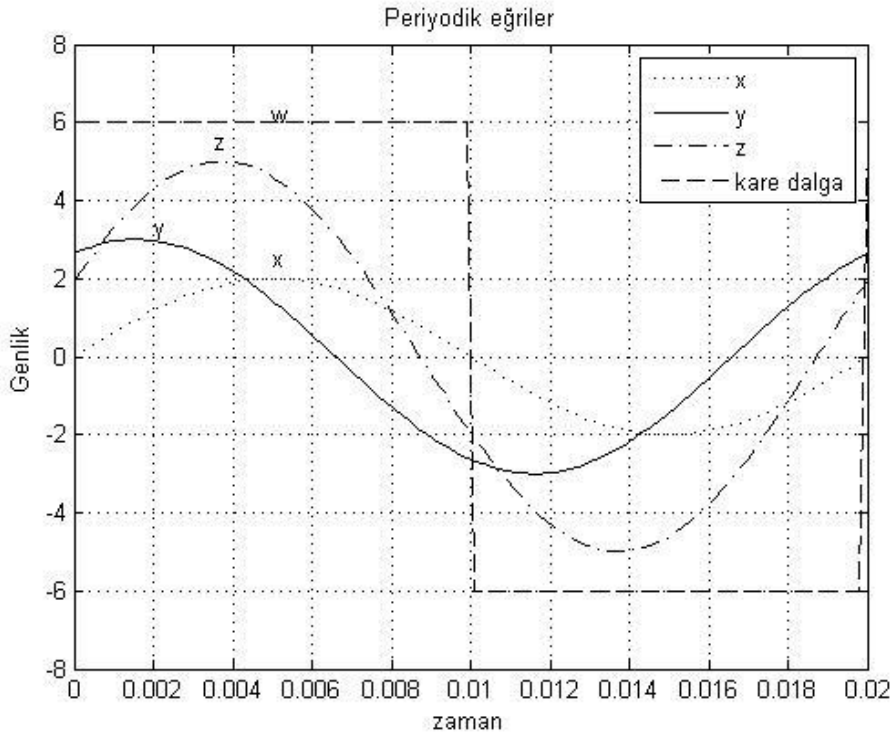
θ: x(t) eğrisinin coswt eğrisine göre faz farkıdır. θ >0 ise x(t) eğrisi coswt eğrisine göre θ açısı kadar ileri fazda, θ <0 ise x(t) eğrisi coswt eğrisine göre θ açısı kadar geri fazdadır denir.

Problem 5.1

$x(t) = 2 \sin 2\pi 50t$
$y(t) = 3 \cos (2\pi 50t - 0.5)$
$z(t) = 5 \sin (2\pi 50t + 0.4)$
ve
genliği 6 birim, frekansı 50 Hz olan **kare dalga eğrilerini t:**[0:0.02:100] saniye aralığında çiziniz.

Çözüm

$x(t) = 2 \sin 2\pi 50t$
x(t) eğrisinin genliği; A=2, x(t) eğrisinin frekansı;
$wt = 2\pi ft = 2\pi 50t \rightarrow f=50$ Hz→ periyodu: T=1/f=0.02 s
y(t) eğrisinin cos(wt) eğrisi ile arasındaki faz farkı;θ=0.5 radyan (geri faz),y(t) eğrisinin genliği;A=3,y(t) eğrisinin frekansı;
$wt = 2\pi ft = 2\pi 50t \rightarrow f=50$ Hz→ periyodu: T=1/f=0.02 s
z(t) eğrisinin sin(wt) eğrisi ile arasındaki faz farkı radyan olarak;θ=0.4 radyan (ileri faz),z(t) eğrisinin genliği;A=5,z(t) eğrisinin frekansı;
$wt = 2\pi ft = 2\pi 50t \rightarrow f=50$ Hz→ periyodu: T=1/f=0.02 s
z(t) eğrisinin sin(wt) eğrisi ile arasındaki faz farkı;θ=0.4 radyan olur. x(t), y(t), z(t) ve kare dalga eğrileri aşağıda verilen programa göre çizilebilir. Programın çalıştırılması sonucunda elde edilen çizim şekil 5.1'de gösterilmiştir.



Şekil 5.1

```
%Periyodik eğrilerin çizimi
t=linspace(0,0.02,100);
x=2*sin(2*pi*50*t);
y=3*cos((2*pi*50*t)-0.5);
z=5*sin(2*pi*50*t+0.4);
w=6*square(2*pi*50*t); % square: periyodik kare dalga çizim
komutudur
plot (t,x, 'k: ' ,t,y, 'k-' ,t,z, 'k-. ' ,t,w, 'k--' );
axis([0   0.02 -8   8]);title('Periyodik eğriler');
ylabel('Genlik'),xlabel('zaman');
grid,legend('x','y','z', 'kare dalga');text(0.005,2.5,'x');
text (0.002, 3.3, 'y') ; text (0.0035,5.5, 'z') ; text (0.005,
6.2,'w');
```

Yukarıda verilen programda text komutu, çizilen eğrilerin takibini kolaylaştırmak için yerleştirilmiştir, text komutunu yazmadan önce bu satırların üzerindeki program çalıştırılmak, eğrilerin konumu görüldükten sonra text komutunun devamına yazılan koordinatlar belirlenmelidir.

5.2. MATLAB ortamında polinom gösterimi

Polinom bir değişkenli bir fonksiyondur ve 'n.' dereceden bir polinomun genel gösterimi;

$$A(x)= a_1x^n + a_2x^{n-1} + a_3x^{n-2} + \ldots\ldots\ldots + a_nx + a_{n+1} \qquad (5.3)$$

olarak verilir. (5.3) ifadesinde 'x' değişkeni tek bir **sayı** olabileceği gibi bir **matris** de olabilir. Polinomun standart gösterimi;

$$A(x)= a_nx^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots\ldots\ldots + a_1x + a_0 \qquad (5.4)$$

olmakla birlikte MATLAB program alt yapısı daha sonra açıklanacağı gibi polinomun (5.4) şeklinde gösterilmesi daha uygundur. Örnek olarak;

$B(x)=3x^4-7x^2+2x-1$

şeklinde 4. dereceden bir polinom göz önüne alınsın. Bu eşitlikte;

- eğer 'x' daha önce tanımlanmış bir **sayı** ise B(x) eşitliği MATLAB ortammda;

```
>> B=3*x^4-7*x^2+2*x-1;
```

olarak yazılır.

- eğer V daha önce tanımlanmış bir **vektör** ya da bir **matris** ise B(x) eşitliği MATLAB ortamında;

```
>> B=3*x.^4-7*x.^2+2*x-1;
```

olarak yazılır. Bu ifadede B ile x aynı boyutta olacaktır.

B=polyval (a, x) : Bir B polinomu MATLAB ortamında polyval komutu ile tanıtılır. Bu komutta x bir sayı olabileceği gibi bir vektör de olabilir. Bu komutta x ile verilen değer veya değer aralığı için, katsayıları a olan bir polinomun aldığı değerler B'ye atanır, a; B polinomundaki katsayıları içeren **vektör** matristir, a vektör matrisinin ilk elemanı yani a(1); (5.3) eşitliğindeki a_1 katsayısıdır. Bu açıklamaya bakılarak neden (5.3) eşitliğindeki katsayı

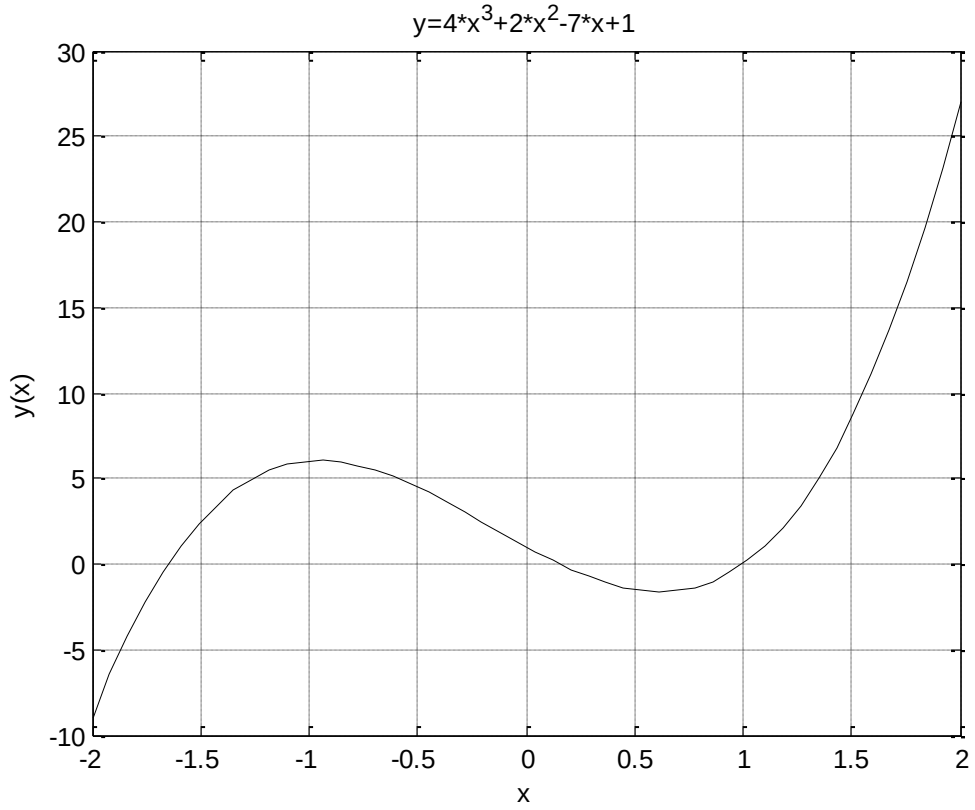
dizilişi yerine (5.4) ifadesindeki dizilişin (MATLAB yazılımında) tercih edildiği anlaşılmaktadır. 'B=polyval(a,x)' ifadesinde kullanılan x değeri ise sayı (ör: x=2) olabileceği gibi bir **vektör** (ör: x=-5:0.1:15 veya x=linspace(-5,15,201) de olabilir. B vektörü ile x vektörü aynı boyutta olacaktır.

Problem 5.2

$y(x) = 4x^3 + 2x^2 - 7x + 1$

polinomunu MATLAB editöründe x:(-2,2) aralığında polyval komutu yardımı ile çizdiriniz.

Çözüm



Şekil 5.2

```
x=linspace (-2,2,50);
a=[4 2 -7 1];
y=polyval (a,x);
plot (x,y,'k-');
title ('y=4*x^3+2*x^2-7*x+1'),
xlabel ('x'), ylabel ('y(x)'),grid
```

Yukarıda verilen MATLAB programının çizimi şekil 5.2' de verilmiştir.

5.2.1. İki polinomun toplamı veya farkı

İki polinomun toplamı (veya farkı) polinomların katsayılarının toplamına (veya farkına) eşittir. İki polinomun MATLAB ortamında toplanabilmesi veya çıkartılabilmesi için katsayıları içeren her iki vektörün**boyutları mutlaka aynı olmalıdır.**

$A(x) = x^4 - 4x^3 - 2x + 6$

$B(x) = 6x^5 - 2x^3 - 5x^2$

olarak verilen iki polinomun toplamı;

$C(x) = A(x)+B(x)$

$C(x) = 6x^5 + x^4 - (4+2)x^3 - 5x^2 - 2x + 6 = 6x^5 + x^4 - 6x^3 - 5x^2 - 2x + 6$

olur. Yukarıda verilen polinom toplamı MATLAB ortamında yapılırsa;

```
>> a=[0 1 -4 0 -2 6];

>> b=[6 0 -2 -5 0 0 ];

>> c=a+b

c =

     6     1    -6    -5    -2     6
```

sonucu elde edilir. c vektörüne bakarak aşağıdaki sonuç yazılabilir;
$C(x) = 6x^5 + 4x^4 - 6x^3 - 5x^2 - 2x + 6$

5.2.2. Polinomun bir sayı ile çarpılması

Bir polinomun bir sayı ile çarpımı bu polinomun katsayılarının ayrı ayrı bu sayı ile çarpılması anlamına gelir.

$A(x) = 2x^4 + 3x^3 - 7x^2 - 2x + 1$ polinomu 2 ile
çarpılırsa;
$B(x)=2*A(x)= 4x^4 + 6x^3 - 14x^2 - 4x + 2$
elde edilir. Bu işlem MATLAB ortamında yazılırsa;

```
>> a=[2 3 -7 -2 1];
>> b=2*a
b =4     6   -14    -4     2
```

olacaktır. Bu sonuca göre;

$$B(x) = 4x^4 + 6x^3 - 14x^2 - 4x + 2$$

elde edilir.

5.2.3. İki polinomun birbiri ile çarpımı

İki polinomun çarpımı için conv komutu kullanılır.

conv(x,y) : ‘x’ve ‘y’ adlı iki vektörün çarpımım hesaplar ve elde edilen polinomun katsayılarını (yine vektör olarak) verir. Burada kullanılan ‘x’ ve ‘y’ vektörleri iki ayrı polinomun katsayılarıdır ve çarpılabilmeleri için **boyutları aynı olmak zorunda değildir.** Aşağıdaki örnek incelenmelidir. Kullanıcının aklına şu soru gelebilir: İki polinom toplanırken boyut eşitliği isteniyor ama çarpımda boyut eşitliği istenmiyor? Bunun cevabı; toplama için özel bir komut olmamasına karşın, arpım işlemi için **komut** (conv) olmasıdır.

$K(x)= 4x^4 - 8x^3 + 7x + 1$
$M(x)= 2x^5 + 7x^3 - 2x + 6$

olarak verilen iki polinomun çarpımı;

$T(x)=K(x)*M(x)=(4x^4-8x^3+7x+1)*(2x^5+7x^3-2x+6)$
$T(x)=8x^9-16x^8+28x^7-42x^6-6x^5+89x^4-41x^3-14x^2+40x+6$

olur. Command window ortamında çarpım işlemi yapılırsa;

```
>> k=[2 3 -4 7];
>>m=[1 0 -3 0 -5 0];
>> t=conv(k,m)
t=8    -16     28-42-689-41-14406
```

elde edilir.

5.2.4 Büyük dereceli polinomun küçük dereceli polinoma bölümü

Büyük dereceli polinomun küçük dereceli polinoma bölümü için deconv komutu kullanılır.

[bolum,kalan] =deconv(pay, payda): pay ve payda adlı iki vektörün bölümünü hesaplar. Bu işlem sonunda, bölüm b ve kalan r vektörlerini verir. Burada kullanılan pay ve payda vektörleri iki ayrı polinomun katsayıları, bolum ve kalan ise sırası ile bölüm ve kalan polinomların katsayılarını içeren vektörlerdir. İki vektörün **pay ve payda boyutları aynı olmak zorunda değildir** (zira bölme için komut var).

Aşağıdaki örnek incelenmelidir;
$X(t)=7t^6-2t^4+8t^3+2$
$Y(t)=2t^4-4t^3+2t+8$
X(t)/ Y(t) değeri bulunmak istenirse;

$$\frac{X(t)}{Y(t)}=\frac{7t^6-2t^4+8t^3+2}{2t^4-4t^3+2t+8}=B(t)+\frac{R(t)}{Y(t)}=3.5t^2+7t+13+\frac{53t^3-42t^2-82t-102}{2t^4-4t^3+2t+8}$$

elde edilir. Bölüm işlemi command window ortamında yapılırsa;

```
>> pay=[7 0 -2 8 0 0 2];
>> payda=[2 -4 0 2 8];
>> [bolum,kalan]=deconv(pay,payda)
bolum =3.5000    7.0000   13.0000

kalan =0     0     0    53   -42   -82  -102
```

bulunur. Sonuçlar X(t)/ Y(t) sonuçları ile karşılaştırılmalıdır.

5.2.5 Polinom türevinin yapılması

Polinomların türevine ilişkin işlemler için polyder komutu kullanılır, polyder komutun birden çok işlemde kullanımı söz konusudur. Bunlar aşağıda uygulamalı olarak gösterilmiştir:

polyder(a): a vektörünün türevini hesaplar.
Aşağıdaki örnek incelenirse;
$S(t)=2t^6+7t^3-8t+3$
polinomunun t'ye göre türevi;

$$\frac{dS(t)}{dt}=\frac{d(2t^6+7t^3-8t+3)}{dt}=12t^5+21t^2-8$$

bulunur. Yukarıdaki işlem MATLAB command window ortamında yapılırsa;

```
>> s=[2 0 0 7 0 -8 3];
>> a=polyder(s)
a =
    12     0     0    21     0    -8
```

elde edilir.
polyder(x,y): x ve y vektörlerinin çarpımının türevini hesaplar.
$x(t)= t^4 +7t^2 -6t +1$
$y(t)= 2t^3 +4t^2 - 3t +8$
$z(t)= x(t) * y(t)$
olduğuna göre,

$$\frac{d}{dt}z(t) = y(t)*\frac{d}{dt}x(t) + x(t)*\frac{d}{dt}y(t) = 14t^6 + 24t^5 + 55t^4 + 96t^3 - 129t^2 + 156t - 51$$

bulunur. Yukarıdaki işlem Command Window ortamında yapılırsa;

```
>> x=[1 0 7 -6 1];
>> y=[2 4 -3 8];
>> Z=polyder(x,y)

Z =

    14    24    55    96  -129   156   -51
```

edilir. Sonuç yukarıdaki dz(t)/ dt türev sonucu ile karşılaştırılmalıdır.
[pay, payda] =polyder {x,y) : x ve y vektörlerinin bölüm türevini hesaplar, pay; türev sonucuna ilişkin kesrin pay vektörünü, payda ise payda vektörünü verir.
$x(t) = 4t^5 - t^4 - 8t^3 + 2t + 9$
$y(t) = 3t^3 - 7t^2 + 3t + 8$
olduğuna göre, X(t)/Y(t)'intürevi hesaplansın:

$$\frac{N(t)}{D(t)} = \frac{d}{dt}\left(\frac{x(t)}{y(t)}\right) = \frac{y(t)*\frac{d}{dt}x(t) - x(t)*\frac{d}{dt}y(t)}{(y(t))^2} = \frac{24t^7 - 87t^6 + 62t^5 + 207t^4 - 92t^3 - 259t^2 + 126t - 11}{9t^6 - 42t^5 + 67t^4 + 6t^3 - 103t^2 + 48t + 64}$$

bulunur. Yukarıdaki işlem Command Window ortamında yapılırsa;

```
>> x=[2 3 -4 7];
>> y=[1 0 -3 0 -5 1];
>> [pay,payda]=polyder(x,y)

pay =

-4    -9    16   -26   -44    54     6    31

payda =

1     0    -6     0    -1     2    30    -6    25   -10     1
```

elde edilir.

5.2.6 Polinom integralinin alınması

polyint(a): Katsayıları a vektörü ile verilen (A) polinomun integralini hesaplar.
Örnek olarak; A= $2t^3+3t^2-5t+6$ polinomunun integrali hesaplanmak istensin:

```
>> a=[2 3 -5 6];
>> polyint(a)

ans =

    0.5000    1.0000   -2.5000    6.0000         0
```

Yukarıda elde edilen sonuca bakarak A polinomunun integrali;
B= $(2t^3+3t^2-5t+6)dt = 0.5\,t^4 + t^3 - 2.5t^2 + 6t$
olmaktadır. polyint (a) ifadesinde integral sabiti **'sıfır'** kabul edilmektedir. Integral sabitinin 'sıfır' kabul edildiği, yukarıda bulunan çözüm vektörünün son elemanının 'sıfır' olmasından da anlaşılmaktadır.
polyint (a, sabit) : İntegral sabiti sıfırdan farklı bir değerde seçildiğinde, bu komut türü kullanılmalıdır.
Örnek olarak integral sabiti 5 alınırsa;

```
>> a=[2 3 -5 6];
>> polyint(a,5)

ans =

    0.5000    1.0000   -2.5000    6.0000    5.0000
```

elde edilir.

5.2.7 Polinom Köklerinin Bulunması

Bir polinomun köklerinin bulunmasına ilişkinişlemler için roots komutu kullanılır.

roots (a) : a vektörünün köklerini hesaplar. Burada a; verilen polinomun katsayı vektörüdür.

D(x)= $(x+1)^2 = x^2+2x+1$

polinomunun kökleri;

$x^1=-1$; $x^2=-1$

değerleridir. D(x) polinomun kökleri command window ortamında bulunmak istenirse;

```
>> d=[1 2 1];
>> c=roots(d)

c =
    -1
    -1
```

elde edilir. İkinci bir örnek olarak;

F(t)= $t^4-4t^3+6t^2-4t$

polinomunun kökleri de MATLAB Command Window ortamında hesaplanırsa;

```
>> f=[1 -4 6 -4 0];
>> k=roots(f)

k =
         0
    2.0000
    1.0000 + 1.0000i
    1.0000 - 1.0000i
```

elde edilir. Yukarıda da görüldüğü gibi dört adet kökten ikisi sanal, ikisi reel köktür. Yukarıda bulunan köklere göre F(t) tekrar yazılırsa;

F(t)=(t-2) (t-1-i) (t-1+i) t

bulunur. Yukarıda bulunan köklerin, F(t) polinomunu sağlayıp sağlamadığı kontrol edilmek istenirse;

```
>> polyval(f,k)

ans =

  1.0e-013 *

         0
    0.1243
   -0.0178 + 0.0355i
   -0.0178 - 0.0355i
```

işlemi yapılabilir. Değerlerinin tam olarak sıfır olmaması, MATLAB hesaplama ortamının

10^{-13} duyarlılıkta işlem yapabilme kapasitesinden kaynaklanmaktadır.

5.2.8. Kökleri bilinen bir polinomun elde edilmesi

Köklerini kullanarak bir polinomu elde etmek için poly komutu kullanılır.

poly{a) : a vektörünün elemanlarını kök kabul eden polinomu bulur. Aşağıdaki örnek incelenmelidir.

Problem 5.3

$x^1=0$, $x^2=2$, $x^3=1+j$, $x^4=1-j$
değerlerini kök kabul eden polinomu bulunuz.

Çözüm

```
>> f=[0 2 1-j 1+j];
>> k=poly(f)

k =
```

```
     1     -4      6     -4      0
```

Yukarıda bulunan k vektörünün elemanlarını katsayı vektörü olarak kabul eden polinom;
$F(t)= t^4 -4t^3 +6t^2 -4t$
olarak bulunur. Yukarıdaki program satırları yerine aşağıdaki komut satırı da yazılabilirdi:

```
>> k=poly([0 2 1-j 1+j])

k =

     1     -4      6     -4      0
```

Problem 5.4

A(x)= $x^3 - 3x^2 + 5x + 3$
B(x)= $4x^4 + 3x^3 - 2$
Yukarıda verilen iki adet polinomun çarpımlarının türevini sıfır yapan x değerleri için A(x) v B(x) polinomlarının aldığı değerleri ayrı ayrı bulan MATLAB programı yazınız. Yazılan program içinde aşağıdaki satırlar da bulunacaktır;

```
.....
disp('A polinomunun türev kökü için aldığı değerler')
.....
disp('B polinomunun türev kökü için aldığı değerler')
```

Çözüm

```
clear all
clc
A=[1 -3 4 3]; B=[4 3 0 0 -2];
D=polyder(A,B);
F=roots(D);
disp('A polinomunun türev kökü için aldığı değerler')
G=polyval(A,F);
disp('B polinomunun türev kökü için aldığı değerler')
H=polyval(B,F);
```

5.3 Pay ve paydasında polinom olan kesir ifadesinde köklerinin bulunması

A(t)= $$\frac{X(t)}{Y(t)} = \frac{a_1 t^k + a_2 t^{k-1} + a_3 t^{k-2} + ... + a_k t + a_{k+1}}{b_1 t^m + b_2 t^{m-1} + b_3 t^{m-2} + ... + b_m t + b_{m+1}}$$ (5.5)

A(t)'nin kökleri, Y(t)'yi sıfır yapan değerlerdir. (5.5) ile verilen A(t) kesrinde iki farklı durum söz konusudur:

- m>k olması durumu (paydanın derecesi payın derecesinden büyük)
- k $\geq$ m olması durumu (payın derecesi paydanın derecesinden büyük veya eşit)

5.3.1. m>k için köklerinin bulunması (payda derecesi paydan büyük)

(5.5) eşitliğinde yer alan Y(t) polinomunun köklerinin $k_1, k_2, k_3, ...k_m$ olduğu kabulü ile;

Y(t)= $b_1(t_1-k_1)(t_2-k_2)(t_3-k_3).....(t_m-k_m)$ (5.6)

olarak yazılabilir. Bu durumda (5.5) ifadesi;

A(t)= $\frac{X(t)}{b_1(t_1-k_1)(t_2-k_2)(t_3-k_3).....(t_m-k_m)}$ (5.7)

olacaktır. (5.7) ifadesinin kökleri (k^i değerleri) reel, sanal veya katlı olabilir. Eğer (5.7) ifadesinin kökleri **katlı değil** ise bu ifade;

A(t)= $\frac{r_1}{(t_1-k_1)}+\frac{r_2}{(t_2-k_2)}+.....+\frac{r_m}{(t_m-k_m)}$ (5.8)

olarak da yazılabilir. (5.8)'de kullanılan r^i değerlerine **rezidü** adı verilir. Rezidüler sanal değerler de alabilirler. (5.7) eşitliğinin kökleri ve rezidü değerleri, residue komutu ile bulunabilir;

[rez, kok]=residue(a, b): a; (5.7) eşitliğinde X(t) polinomunun katsayılarını içeren vektör, b; Y(t) polinomunun katsayılarım içeren vektör, rez;(5.8) ifadesindeki rezidü değerleri, kok; (5.8) ifadesindeki kök değerleridir

Command Window ortamında residue komutunun uygulanışını görmek için aşağıdaki örnekler incelenmelidir:

Problem 5.5

A(t)= $\frac{t^2-4}{2t^4-6t^3+2t^2+5}$

probleminin rezidü ve köklerini MATLAB ortamında bulunuz ve A(t) ifadesini (5.8) formunda yazınız.

Çözüm

```
>> pay=[1 0 -4];
>> payda=[2 -6 2 0 5 ];
>> [rez,kok]=residue(pay,payda)

rez =

   0.1063
   0.2580
  -0.1821 + 0.1804i
  -0.1821 - 0.1804i

kok =

   2.4041
   1.4180
  -0.4111 + 0.7512i
```

```
-0.4111 - 0.7512i
```

Yukarıda elde edilen sonuçlara bakarak;

$$A(t)=\frac{0.1063}{(t_1-2.4041)}+\frac{0.2580}{(t_2-1.4180)}+\frac{-0.1821+0.1804i}{(t_3+0.4111-0.7512i)}+\frac{-0.1821-0.1804i}{(t_4+0.4111+0.7512i)}$$

yazılabilir. Yukarıda görüldüğü gibi köklerden iki tanesi reel, diğer ikisi sanal ve eşleniktir. Rezidü ve kök değerlerinin sayısı aynı zamanda çözümün kaç adet kesir içerdiğini de gösterir. Eğer (5.8) ifadesinin köklerinden bazıları **katlı** ise bu ifade;

$$A(t)=\frac{r_1}{(t_1-k_1)}+\frac{r_2}{(t_2-k_1)^2}+.....+\frac{r_p}{(t_p-k_1)^p}+\frac{r_{p+1}}{(t_{p+1}-k_{p+1})}+....+\frac{r_m}{(t_m-k_m)} \qquad (5.9)$$

şeklinde gösterilebilir. (5.9) ifadesine bakarak k_1 kökünün 'p' katlı olduğu söylenebilir. Diğer kökler reel veya sanal değerdedir.

5.3.2. $k \geq m$ için köklerinin bulunması (pay derecesi paydadan büyük)

(5.5) ifadesi $k \geq m$ koşulu altında;

$$A(t)=\frac{X(t)}{Y(t)}=B(t)+\frac{K(t)}{Y(t)} \qquad (5.10)$$

yazılabilir. (5.10) ifadesinde B(t); 'bölüm' polinomu, K(t); 'kalan' polinomu olarak adlandırılır. Bu tür kesirlerde kök, rezidü ve kalan polinom katsayılarını bulmak için residue komutu kullanılır.(5.10) ifadesindeki K(t) polinom derecesinin , Y(t) polinom derecesinden küçük olduğu unutulmamalıdır.

[rez,kok,bolum]=residue(pay,payda): pay, (5.10) eşitliğinde X(t) polinomunun katsayılarını içeren vektör, payda; Y(t) polinomunun katsayılarını içeren vektör, rez; K(t)/Y(t) ifadesindeki rezidü değerleri, kok; K(t)/Y(t) ifadesindeki kök değerleri, bolum; B(t) polinomunun katsayılarım gösteren vektördür.

Problem 5.6

$$A(t)=\frac{t^5-12t^4+31t^3+30t^2-201t+179}{t^4-6t^3-3t^2+52t-60}$$

polinomunun kök, rezidü ve kalan polinom sayısı katsayılarını MATLAB ortamında bulunuz ve bu sonuçlara bakarak A(t) ifadesini (5.10) formunda yazınız.

Çözüm

```
>> pay=[1 -12 31 30 -201 179];
>> payda=[1 -6 -3 52 -60 ];
>> [rez,kok,bolum]=residue(pay,payda)
rez =

   -8.0000
    5.0000
```

```
    1.0000
    1.0000

kok =

    5.0000
   -3.0000
    2.0000
    2.0000

bolum =

  1  -6
```

Yukarıda elde edilen sonuçlara bakarak;

$$A(t) = t - 6 + \frac{1}{t-2} + \frac{1}{(t-2)^2} + \frac{5}{t+3} - \frac{8}{t-5}$$

yazılabilir.

Problem 5.7

$$H_1 = \frac{12s^2 - 3}{s^3 + 4s + 1}; H_2 = \frac{4s^5 - 3s^2}{0.25s + 5}$$

Yukarıda verilen iki polinomun çarpımlarından elde edilen polinomun rezidü ve köklerini bulan MATLAB programını yazınız.

Çözüm

```
>> carpim_pay=conv([12 0 -3],[4 0 0 -3 0 0]);
>> carpim_payda=conv([1 0 4 1],[0.25 5]);
>> [rezidu,kok,katsayi]=residue(carpim_pay,carpim_payda)
rezidu =

  1.0e+007 *

   3.0403
   0.0000 + 0.0000i
   0.0000 - 0.0000i
   0.0000

kok =

 -20.0000
   0.1231 + 2.0113i
   0.1231 - 2.0113i
  -0.2463

katsayi =

         192       -3840       75984    -1520016
```

5.3.3 Kök, rezidü ve kalan polinom katsayıları verildiğinde pay ve payda polinomunun elde edilmesi

$$A(t)=\frac{X(t)}{Y(t)}=B(t)+\frac{R(t)}{Y(t)}$$

ifadesinde pay ve paydaşında polmom olan bir kesirde kök, rezidü ve kalan polinom katsayılarının nasıl bulunduğu gösterildi. Burada ise aynı ifadede köle, rezidü ve kalan polinom katsayıları verildiğinde pay ve payda polinomunun (X(t)/Y(t)) nasıl elde edileceği gösterilecektir. Bu amaçla kullanılan komut yine residue ifadesidir;

[pay, payda] = residue(rez,kok,bolum): pay; X(t) polinomunun katsayılarını içeren vektör, payda; Y(t) polinomunun katsayılarını içeren vektör, rez; R(t)/Y(t) ifadesindeki rezidü değerleri, kok; R(t)/Y(t) ifadesindeki kök değerleri, bolum; B(t) polinomunun katsayılarını gösteren vektördür.

Problem 5.8

$$A(t)=t+2-\frac{0.5}{t+4}-\frac{0.6145}{t-1}+\frac{1.5}{t-7}$$

polinomunu;

$$A(t)=\frac{X(t)}{Y(t)}$$

formunda yazınız.

Çözüm

```
>> rez=[-0.5 -0.6145 1.5];
>> kok=[-4 1 7];
>> bolum=[1 2];
>> [pay,payda]=residue(rez,kok,bolum)

pay =

    1.0000   -2.0000  -32.6145  -11.6565   63.7060

payda =

     1    -4   -25    28
```

Yukarıdaki sonuçlara bakarak;

$$A(t)=\frac{t^4-2t^3-32.6145t^2-11.6565t+63.7060}{t^3-4t^2-25t+28}$$

yazılabilir. Eğer bolum polinomu olmasaydı; bolum= [ ] olacaktı.

Problem 5.9

$$H(s)=\frac{A(s)}{B(s)}=s-4+\frac{20.6145}{s+4.4495}-\frac{0.6145}{s-0.4495}$$

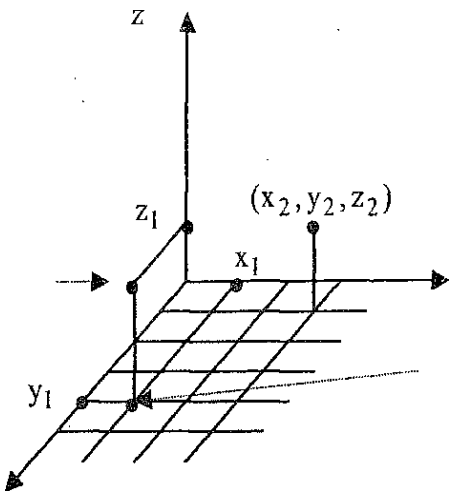
denklemini sıfır yapan s değerlerini bulan MATLAB programı yazınız.

Çözüm

```
>> rezidu=[20.6145 -0.6145];
>> kok=[-4.4495 0.4495];
>> bolum=[1 -4];
>> [pay,payda]=residue(rez,kok,bolum);
>> roots(pay)%A(s)'nin kök değerleri
```

5.5. Üç boyutlu yüzey ve eğri çizimi

mesh, meshc, meshz, meshgrid, surf, surfc contour gibi komutlar MATLAB ortamında üç boyutlu yüzey çizimi için kullanılırlar. plot3 komutu ise üç boyutlu eğri çizimi için kullanılır.



Şekil 5.3

z=f(x,y) fonksiyonunu çizmek için eğrinin geçtiği noktalar belirlenmelidir. Şekil 5.3'de görüldüğü gibi $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots$ noktaları yüzeyin belirlenmesi açısından çok önemlidir. (x_1, y_1) koordinatları f(x,y) eşitliğinde yerlerine konulduklarında f(x_1, y_1)=z_1 noktası olacaktır. Benzer şeyler f(x_2,y_2)=z_2 noktası için de söylenebilir.

Üç boyutlu bir yüzey çizimi yapmak için bu çizimin hangi aralıklarda yapılacağı belirlenmelidir. Örneğin x ekseni üzerindeki değişim -5 ile 5 arasında, y eksenindeki değişim 6 ile 12 arasında gerçekleştirilecek ise, bu aralıklar arasında yer alan her (x_i, y_i) koordinatlarına karşı gelen z_i değerleri de hesaplanmalıdır. Üç boyutlu çizim ancak bundan sonra gerçekleştirilebilir. x: [-5 5] aralığı ve y: [6 12] aralığındaki x ve y vektör boyutu arttıkça çizimin hassasiyeti de artar. x_i ve y_i değerleri MATLAB ortamında vektörler ile tanımlanır. Bu vektör değerleri yardımı ile z_i

değerleri bulunmaya çalışılır. Aşağıdaki command window satırlarına bakılarak x ve y vektörlerinin nasıl oluşturulduğu görülmelidir.

```
>> x=-pi:2.8584

x =

   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584

>> y=0:0.2:1

y =

         0    0.2000    0.4000    0.6000    0.8000    1.0000
```

x ve y vektörlerinden üç boyutlu yüzey çizimi için gerekli (ızgaralar) X ve Y matrislerini elde etmek üzere meshgrid komutu kullanılır. Bu komut hedefe ulaşmada kullanılan ara bir komuttur.

[X,Y]=meshgrid(x,y) : x ve y vektörlerini üç boyutlu yüzey çizimi yapabilmek için X ve Y matrislerine dönüştürerek ızgara şeklinde çizim yüzeyi meydana getirir. Bu komut ile x vektörünün tüm elemanları X matrisinin satırlarını oluşturur (tüm satırlar aynı olur). X matrisinin satır sayısını ise y vektörünün boyutu belirler. Bu komut aynı zamanda y vektörünün elemanları ile Y matrisinin sütunlarım oluşturur (tüm sütunlar aynı olur). Y matrisinin sütun sayısını x vektörünün boyutu belirler. x; n boyutlu, v; m boyutlu ise X ve Y matrisleri (m*n) boyutundadır.

Aşağıdaki örnek incelenmelidir:

```
>> x=-pi:2.8584

x =
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584

>> y=0:0.2:1

y =
         0    0.2000    0.4000    0.6000    0.8000    1.0000

>> [x,y]=meshgrid(x,y)

x =
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584
   -3.1416   -2.1416   -1.1416   -0.1416    0.8584    1.8584

y =
```

```
         0         0         0         0         0         0
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000
    0.4000    0.4000    0.4000    0.4000    0.4000    0.4000
    0.6000    0.6000    0.6000    0.6000    0.6000    0.6000
    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

Not: Yukarıda verilen program satırlarında x ve y vektörleri meshgrid komutu içinde de tanımlanabilirdi:

```
>> [x,y]=meshgrid(-pi:2.8584,0:0.2:1);
```

Problem 5.10

x = -pi:2.8584, y =0:0.2:1 aralığında z = f (x, y) = cos(x^2 - y) eğrisinin aldığı değerleri bulunuz.

Çözüm

```
>> [x,y]=meshgrid(-pi:2.8584,0:0.2:1);
>> z=cos(x^2-y)

z =
    0.8904   -0.3804   -0.3124    0.8551   -0.9868    0.6442
    0.7822   -0.1891   -0.4949    0.9411   -0.9349    0.4794
    0.6428    0.0098   -0.6577    0.9895   -0.8457    0.2955
    0.4778    0.2083   -0.7942    0.9985   -0.7228    0.0998
    0.2938    0.3984   -0.8991    0.9677   -0.5711   -0.0999
    0.0980    0.5727   -0.9682    0.8983   -0.3967   -0.2956
```

Z matrisinin satır sayısının y vektörü boyutunda, sütun sayısının ise x vektörü boyutunda olduğu unutulmamalıdır. Yukarıda verilen programda Z(l,l) değeri; X(l,l) değeri ile Y(l,l) değerinin f(x,y) fonksiyonunda yerlerine konulması sonucunda elde edilir. Benzer şekilde örneğin X(2,3) elemanı ile Y(2,3) elemanı f(x,y) ifadesinde yerine konulursa Z(2,3) elemanı elde edilir. meshgrid komutu üç boyutlu çizim için **mutlaka** yapılması gereken **bir ön adımdır.**

5.5.1. Üç boyutlu yüzey çizim komutları

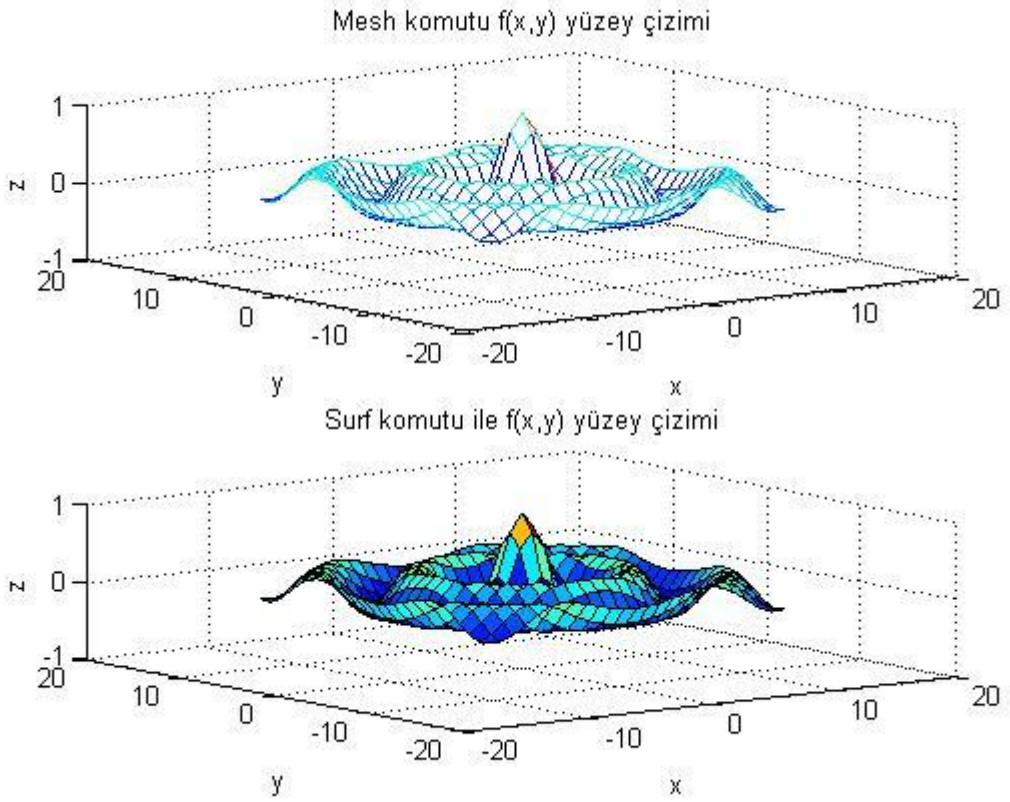
mesh(X,Y,Z): X (yerine x vektörü de konulabilir) ve Y (yerine y vektörü de konulabilir) matrislerindeki değerlerden Z fonksiyon değerlerini üreterek üç boyutlu yüzey çizimi yapar. Bu çizimde ızgara aralıkları **boştur** (şekil 5.6-üst şekil).

surf(X,Y,Z): X (yerine x vektörü de konulabilir) ve Y (yerine y vektörü de konulabilir) matrislerindeki değerlerden Z fonksiyon değerlerini üreterek üç boyutlu yüzey çizimi yapar. Bu çizimde ızgara aralıkları **doludur** (şekil 5.6-alt şekil).

Not: meshgrid komutu x ve y vektörleri ile belirtilen alam X ve Y dizilerine dönüştürür. Böylece 3 boyutlu yüzey çizimi için bir **alt yapı** oluşturur. mesh komutu ise Z=f(X,Y) iki değişkenli fonksiyonun belirttiği yüzeyin ızgara şeklinde grafiğini çizdirir. Bu nedenle meshgrid komutunun 3 boyutlu yüzey çizimi için **ön bir adım** olduğu iyi anlaşılmalıdır. Aşağıda verilen örnek incelenmelidir;

Problem 5.11

x =-12 :12, y =-12:12 aralığında z = $j_0\sqrt{x^2+y^2}$, üç boyutlu yüzeyini hem mesh hem de surf komutu ile çizdiren MATLAB programım yazımz. (J_0: Bessel fonksiyonu)



ŞEKİL 5.4

Çözüm

```
[X Y]=meshgrid(-12:12,-12:12);
r=sqrt(X.^2+Y.^2);
Z=bessel(0,r); subplot (2,1,1), mesh(X,Y,Z) ;
title('Mesh komutu f(x,y) yüzey çizimi');
xlabel('x');ylabel('y');zlabel('z');subplot(2,1,2);surf(X,Y,Z);
title('Surf komutu ile f(x,y) yüzey çizimi'),
xlabel('x') ;ylabel('y'),zlabel('z');
```

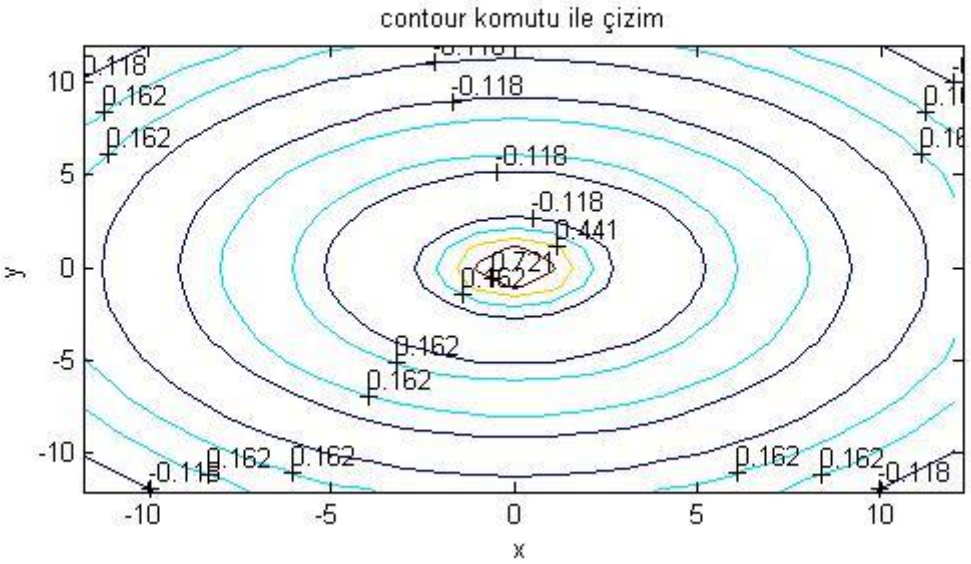
Yukarıda verilen MATLAB programından elde edilen şekil penceresi şekil 5.4'de gösterilmiştir.

contour (x, y, z): x vektörü x eksenindeki çizimin aralığını, y vektörü y eksenindeki çizim aralığını gösterir. z ise x ve y vektörlerinden hareketle fonksiyonun aldığı değerleri gösteren matristir. contour komutu özellikle haritacılıkta eşyükselti eğrilerini elde etmek için kullanılır. Burada kullanılan eğri sayısı otomatik olarak seçilir.

Aşağıdaki örnek incelenmelidir;

```
[X Y]=meshgrid(-12:12,-12:12);
```

```
r=sqrt(X.^2+Y.^2);
Z=bessel(0,r); subplot (2,1,1), mesh(X,Y,Z) ;
c=contour(X,Y,Z,4);
clabel(c) % contour yuksekligi ilave ediliyor
title('contour komutu ile çizim');
xlabel('x');ylabel('y');zlabel('z');
```



Şekil 5.5

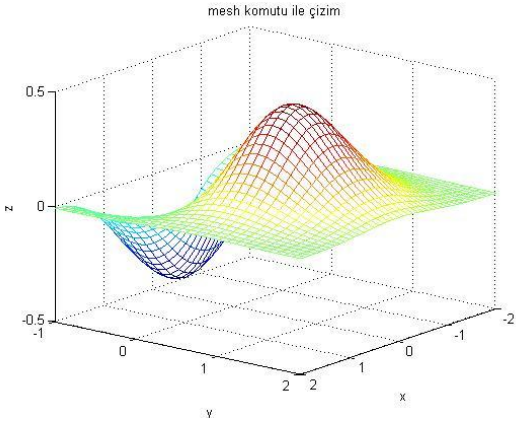
contour(x,y;Z,v) : contour komutu özellikle haritacılıkta kullanılan eşyükselti eğrilerini elde etmek için kullanılır. x vektörü x eksenindeki çizimin aralığını, y vektörü y eksenindeki çizim aralığım gösterir. Zise x ve y vektörlerinden hareketle fonksiyonun aldığı değerleri gösteren matristir. Burada kullanılan v değeri (toplam) eğri sayısını verir. clabel (c) komutu (x,y) düzlemi üzerindeki her eğri üzerine o eğrinin yüksekliğini belirten rakam yerleştirir (bkz.şekil 5.5).

meshc (X,Y,Z) : X (yerine x vektörü de konulabilir) ve Y (yerine y vektörü de konulabilir) matrislerindeki değerlerden z fonksiyon değerlerini üreterek üç boyutlu yüzey çizimi yapar. Bu çizimde ızgara aralıkları boştur. Bu komut ile çizilen üç boyutlu yüzeyin altında 'eşyükselti eğrileri' de çizilir. Böylece mesh yerine meshc komutu kullanılarak yüzeyin x-y düzlemi üzerindeki izdüşümünün görüntülenebilmesi mümkün olabilmektedir. mesh yerine meshz komutu kullanılır ise yüzeye (z ekseni) derinlik katılarak değişim hacimsel olarak görüntülenebilir.

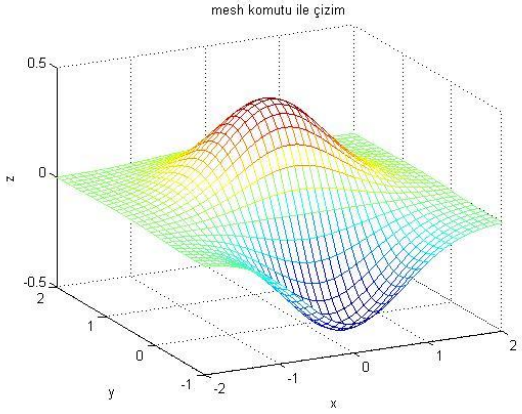
Aşağıdaki program satırları incelenmelidir:

```
x=-2:0.1:2; y=-1:0.1:2;
[X Y]=meshgrid(x,y);
Z=Y.*exp(-(X.^2+Y.^2));
mesh(x,y,Z);
title('mesh komutu ile çizim');
xlabel('x');
ylabel('y');zlabel('z');
```

Yukarıda verilen program satırlarının uygulanması sonunda elde edilen görüntü şekil 5.6'da verilmiştir.

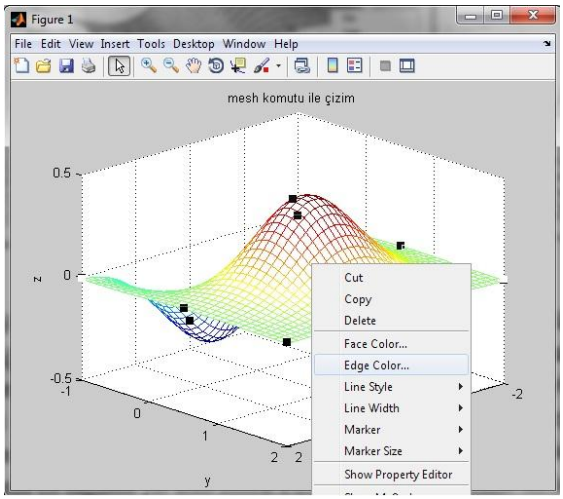


Şekil 5.6

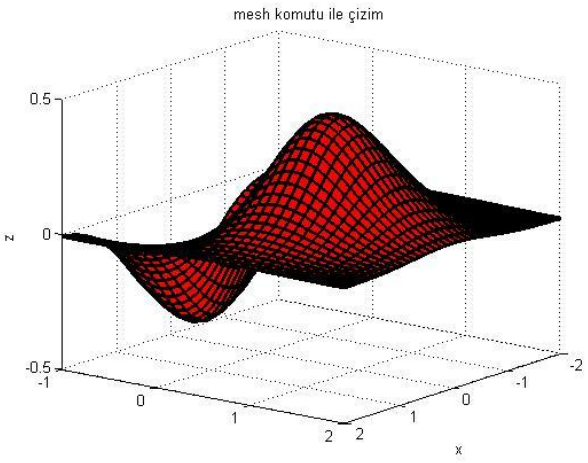


Şekil 5.7

Şekil 5.6'da ikon menüsü içinde yer alan rotate ikonuna 'tık'lanıldığında üç boyutlu yüzeye istenilen açıdan bakılabilir. Şekil 5.6 'ya böyle bir işlem uygulandığında şekil 5.7'deki görüntü elde edilebilir.



Şekil 5.8



Şekil 5.9

Şekil 5.6'da ikon menüsü içinde yer alan Edit Plot (ok işareti) üzerine 'tık'lanıldığında yüzey, üzerinde değişikliğe hazır hale gelir. Eğer fare yüzey üzerindeyken sağ tuşuna 'tık'lanıldığında şekil 5.8'de gösterilen alt pencere açılır. Bu pencere içinde Edge Color seçeneği ile yüzeyin çizgi renklerini, Face Color seçeneği ile de yüzey üzerindeki beyaz renkli boşlukların rengi değiştirilebilir. Line Width ile yüzeyi oluşturan çizgilerin kalınlığını, Line Style seçeneği ile çizginin türünü belirlemek mümkündür. İkon menusunun en sağında yer alan Show Property Editor . . ikonuna 'tık'lanıldığında ise Bölüm 4'de de anlatıldığı yeni bir pencere daha açılır. Bu pencere yardımı ile şekil üzerinde bir çok değişiklik gerçekleştirilebilir.

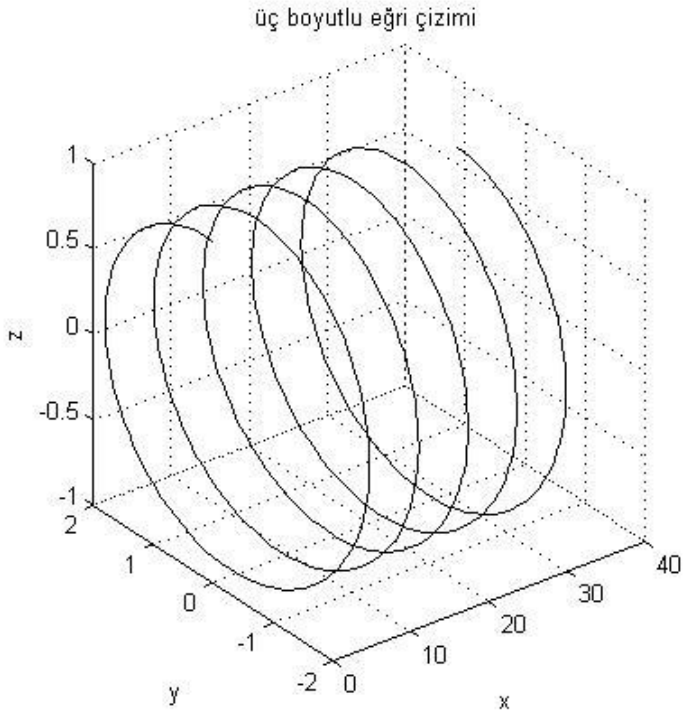
Eğer kullanıcı yüzeyin çok hassas noktalarını (örneğin en yüksek noktasının koordinatlarını) bilmek istiyor ise önce ikon menüsü içinde yer alan Zoom In (+ işaretli büyüteç) ikonuna 'tık'layarak bu ikonu aktif hale getirmelidir. Böylece istediği hassas bölgeye yakınlaşabilir. Daha sonra Pan (beş parmak işareti) ikonuna 'tık'layarak bu ikonu aktif hale getirmelidir. Bundan sonra fareyi hassas bölgeye getirip sol tuşuna basarak (ve basılı tutarak) yüzeyi istediği noktaya

çekebilir. Bu adımı da tamamladıktan sonra Data Cursor ikonuna ‘tık’layarak aktif hale getirip, fare ile yüzey üzerinde hassas noktaya 'tıklamalıdır. Tüm bu işlemler sonunda elde edilen görüntü şekil 5.9'da gösterilmiştir.

5.5.2. Üç boyutlu eğri çizim komutu

x,y ve z eksenlerindeki tüm noktalan belirli olan üç boyutlu bir eğrinin çizimi için plot3 komutundan faydalanılabilir. Aşağıda x, y ve z eksen değerleri verilen üç boyutlu bir eğrinin çizimini yapan bir m dosyası gösterilmiştir.

```
x=0:pi/50:10*pi; y=2*sin(x); z=cos(x);
plot3(x,y,z,'k')
title('üç boyutlu eğri çizimi')
xlabel('x'),
ylabel('y'),zlabel('z')
grid on
axis square
```



Şekil 5.10

Şekil 5.10'da yukarıda verilen m dosyasının çalıştırılması sonunda elde edilen şekil penceresi gösterilmiştir.

5.6. MATLAB ortamında altprogram yapısı

Programlama dillerinde program içinde tekrarlanan bazı işlemler için altprogram mantığı geliştirilmiştir. Bundan amaç ana program içinde tekrarlanan bazı işlemleri tekrarlamamak, bu işlemleri altprogram içine yollayarak orada halletmektir.

MATLAB da altprogram function komutu ile gösterilir. Aşağıda MATLAB Command Window ortamında yazılan satırlar verilmiştir;

```
>> R=[1 2 3 4 5 ];
>> S=alanbul(R)

S =

    3.1416   12.5664   28.2743   50.2655   78.5398
```

Aşağıdaki satırlar ise MATLAB editör ortamında yazılan alanbul. m adlı programın satırlarıdır;

```
function y=alanbul(z)
y=pi*z.^2;
```

Yukarıda iki ayrı bölüm olarak verilen program satırlarının çalışması şu şekilde olmaktadır; MATLAB, x vektörünün ilk değerini (1) almakta bunu alanbul (x) satırına taşımaktadır. alanbul (x), MATLAB için yabancı bir komut olduğundan önce MATLAB dosyalan içinde bu ad verilmiş dosya aranır. Eğer bulunur ise (ve tanımlamalar doğru yapılmış ise) 1, alanbul.m içinde yer alan y=pi*z .^2 ifadesinde kullanılır ve elde edilen sonuç ana programa geri götürülerek ana program içindeki alanbul (x) ifadesinde x yerine konularak 1 yarıçaplı dairenin alanı hesaplanır. Bu işlem sonunda bulunan değer (command window içindeki) anaprogramda S vektörünün ilk elemanı olarak atanır. Daha sonra aynı işlemler x'in diğer değerleri için (2,3....) ayrı ayrı yapılır. Yukarıda verilen alt program, bir giriş (z) ve bir çıkışlı (y) bir programdır.
Aşağıda verilen MATLAB programında alt ve üst taban uzunlukları ve yüksekliği verilen bir yamuğun alan hesabı altprogram kullanılarak yapılmaktadır:

```
>> a=5;%alt taban uzunluğu
>> b=7;%üst taban uzunluğu
>> h=12; %yamuğun yüksekliği
>> alan=hesapla(a,b,h)

alan =
    72
```

Yukarıda verilen ana programın çağırdığı hesapla adlı altprogram aşağıda verilmiştir:

```
function S=hesapla(m,n,k)
S=(m+n)*k/2;
```

Tekrar hatırlanmalıdır ki; altprogram satırları, MATLAB editör ortamında adı 'altprogram adı' ile aynı olan .m dosyası içine yazılmalıdır. Yukarıda verilen örnekte olduğu gibi ana programda geçen sindeg değişkeni alt program adı olarak (sindeg.m veya hesapla.m) editör ortamında oluşturulmuştur. Yukarıda verilen örnek program anlaşıldıktan sonra şimdi function komutunun MATLAB ortamındaki çeşitli kullanış biçimleri tanıtılacaktır;

`function y='f.adı'(x)`: x değeri ana programdan alınır, bu değer altprogram içinde gerekli işlemlerde kullanıldıktan soma y değeri elde edilir. y değeri ise ana programda kullanılır. x ve y değerlerinin ana programdaki karşılıkları farklı olabilir. Eğer ana programın diğer satırlarında (altprogramı çalıştıran komuttan önce)

tesadüfen değişken olarak x ve y kullanılmış ise bu değerler altprogramdaki x ve y değerleri yerine kullanılamazlar. Bu komut tek giriş tek çıkış için kullanılır. Yukarıdaki altprogramı çağıran ana program satırı aşağıda verilmiştir;

```
>> a='f.adı'(b);
```

function y= 'f .adı' (x,z,m. .): x, z, m, . . değerleri ana programdan alınır, bu değerler altprogram içinde kullanıldıktan sonra burada elde edilen y değeri ana programa (ör: a gibi) farklı bir adla gönderilir. Bu komut çok girişli tek çıkışlı altprogram uygulamalarında kullanılır. Yukarıdaki altprogramı çağıran ana program satırı aşağıda verilmiştir;

```
>> a='f.adı'(a,b,c,d,...);
```

function [y, z,m, . .] = 'f. adı' (x) : x değeri ana programdan alınır, bu değer yardımı ile elde edilen y,z,m, .. çıkış değerleri ana programa başka adlar altında atanır (ör:a,b,c,…. gibi). Bu komut tek giriş çok çıkış gerektiren altprogram uygulamalarında kullanılır. Yukarıdaki altprogramı çağıran ana program satırı aşağıda verilmiştir;

```
>> [a b c..]='f.adı'(d);
```

function [y,z,m,..] = 'f .adı' (x,w, t,g, . .): x,w,t,g,.. değerleri ana programdan alınır, bu değerler yardımı ile elde edilen y,z,m, . . çıkış değerleri ana programa başka adlar altında atanır (ör:a,b,c, . . gibi). Bu komut çok giriş çok çıkış gerektiren altprogram uygulamalarında kullanılır.

```
function [y,z,m,..]='fonksiyon adı'(x,w,t,g,..)
…..
…..
```

Yukarıdaki altprogramı çağıran ana program satırı aşağıda verilmiştir:

```
>> [a b c..]='fonksiyon adı'(d,f,h,r,..);
```

Problem 5.12

$y=ax^2+bx+c$
olarak verilen ikinci dereceden bir denklemin katsayıları girildiğinde her iki kökü de altprogram içinde hesaplayan ve Command Window ortamında yazdıran bir MATLAB programı yazınız.

Çözüm

```
>> a=1; b=5; c=6;
>> [x1 x2]=kokbul1(a,b,c)
```

```
x1 =
     -2

x2 =

     -3
```

Yukarıda görülen `kokbul1.m` adlı alt program satırları ise aşağıda verilmiştir.

```
function [h1,h2]=kokbul1(k1,k2,k3)
x=-k2/(2*k1); y=sqrt(k2^2-4*k1*k3)/(2*k1);
h1=x+y; h2=x-y;
```

5.6.1. MATLAB ortamında altprogram içinde altprogram kullanılması

MATLAB ortamında altprogram (ana altprogram) içinde bir veya daha çok sayıda altprogram (yardımcı altprogram) kullanılabilir. Yardımcı altprogramlar ancak ana altprogram tarafından çağrılabilirler. Bunun dışındaki bir yolla yardımcı altprogramlara erişim imkansızdır. Ana altprogram içinde yardımcı altprogram kullanılmasının nedeni yeni altprogram dosyalan (mfile) açmamak, takip edilmesi kolay olan altprogramlar yazabilmektir.

Problem 5.13

(a/al)x2+(b/b1)x+(c/c 1)=0

denkleminde a=l; b=5; c=6 değerlerim almaktadır. a 1, b 1, c 1 değerleri ise;

$3a_1^{-4}=0$; $b_1+5=0$; $2C_1+2=0$

denklemlerini sağlamaktadır. Verilen ikinci dereceden denklemin köklerini bulan MATLAB programını yazınız.

Çözüm

```
>> a=1; b=5; c=6;
>> [x1 x2]=kokbul2(a,b,c)
```

Yukarıda adı geçen kokbul2.m adlı altprogram aşağıda verilmiştir:

```
function [h1,h2]=kokbul2(k1,k2,k3)
w2=[3 -4];
a1=kokbul3(w2);
w3=[1 5];
a2=kokbul3(w3);
w4=[2 2];
a3=kokbul3(w4);
a=k1/a1;
b=k2/a2;
c=k3/a3;
x=-b/(2*a);
y=sqrt(b^2-4*a*c)/(2*a);
h1=x+y;
h2=x-y;
```

```
function u1=kokbul3(n1)
u1=roots(n1);
```

Yukarıda görüldüğü gibi ana altprogram içinde 1 adet yardımcı altprogram kullanılmaktadır. Ana altprogram 16 satırdan, yardımcı altprogram ise 2 satırdan oluşmaktadır. Command Window ortamında verilen 4 satırın çalıştırılması sonunda elde edilen kök değerleri aşağıda gösterilmiştir:

```
>>
x1 =
    3.5726

x2 =
   -2.2393
```

5.7. Tek değişkenli fonksiyonun minimum noktasının bulunması

Alt fonksiyon uygulamasına bir örnek olması için tek değişkenli bir fonksiyonun verilen bir aralıkta minimum değerini bulan bir program yazılacaktır. MATLAB ortamında bu amaçla kullanılan komutlardan bir tanesi de fminbnd komutudur:

x=fminbnd('f .adı',x_1,x_2): Bu komutta 'fonksiyon adı' yazılan yere bilinen (ör.sin,cos,tan..) gibi fonksiyonlar gelebileceği gibi kullanıcı tarafından tanımlanan bir fonksiyon da kullanılabilir. xl yerine fonksiyonun inceleneceği x aralığının alt sınır değeri, x2 yerine de üst sınır değeri yazılır. Bu iki sınır arasında eğriyi minimum yapan değer x olarak hesaplanır. Eğer fonksiyon MATLAB arşivinde tanımlı ise altprogram kullanılmasına gerek olmaz. Eğer fonksiyon kullanıcı tarafından tanımlanmış ise iki tırnak arasına fonksiyonun tanımlandığı altprogramın adı yazılabileceği gibi iki tırnak arasına fonksiyonun kendisi de yazılabilir.

Aşağıdaki örnekler incelenmelidir.

```
>> fminbnd('cos',0.1,5)

ans =

    3.1416
```

Yukarıda verilen MATLAB yazılımında cos(x) fonksiyonunu x=[0,l:5] aralığında minimum yapan x değeri hesaplanmış ve x=3.1416 bulunmuştur.

```
>> x=fminbnd('x.^2-3*x',-2,2)

x =

    1.5000
```

Yukarıda verilen MATLAB yazılımında 'x.^2-3*' fonksiyonunu x=[-2,2] aralığında minimum yapan x değeri hesaplanmış ve x=l. 5 bulunmuştur.

Problem 5.14

$z = 5t^3 - 6t^2 + 8$
fonksiyonunu $-1 \leq t \leq 3$ aralığında minimum yapan t değerini bulunuz.

Çözüm

```
>> tmin=fminbnd('minimumbul',-1,3)
tmin =

    0.8000
```

tmin değeri yaklaşık sıfır alınabilir. Yukarıda ana program satın verilen MATLAB programına ilişkin altprogram ise MATLAB editör ortamında dosya adı minimumbul.m olan . m dosyasma yazılacaktır:

```
function z=minimumbul(t)
z=5*t.^3-6*t.^2+8
```

Yukarıda verilen 2 satır minimumbul.m adlı altprograma ilişkin satırlardır. Eğer verilen bir fonksiyonu tanımlanmış bir aralıkta minimum yapan değere ilave olarak bu fonksiyonun bulunan minimum değer için aldığı değer de merak edilirse aşağıdaki komut kullanılmalıdır:

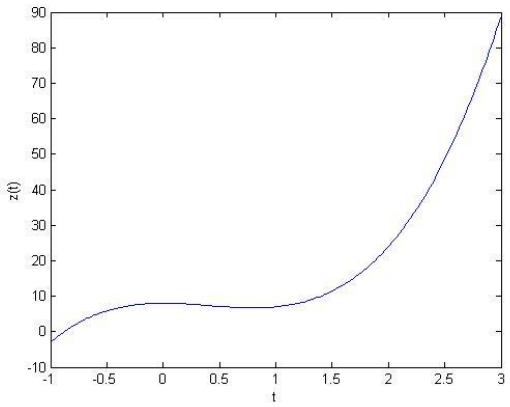
[x,fonkdegeri]=fminbnd ('f .adı' ,xl,x2): Fonksiyonun verilen aralıkta minimum yapan değer için aldığı değer; fonkdeğeri değişkenine atanır, fonkdegeri değişkeninin MATLAB ortamında 'fonksiyonun aldığı değer' anlamına geldiği daha önce açıklanmıştı.

Aşağıdaki örnek incelenmelidir:

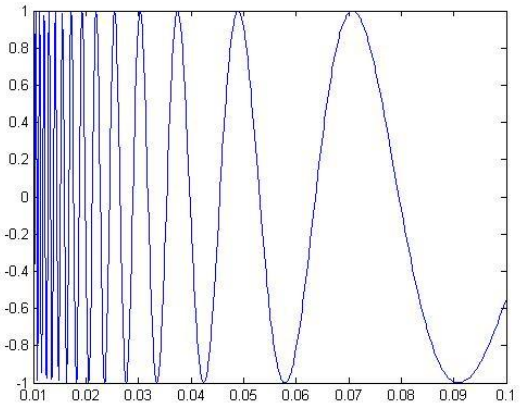
```
>> [tmin,fonkdegeri]=fminbnd('minimumbul',-1,3)
tmin =

    0.8000

fonkdegeri =

    6.7200
```

tmin ve fonkdegeri değerlerinin her ikisi de sıfır kabul edilebilir. Yukarıdaki ana programa ilişkin satırların çalışması için gerekli olan minimumbul.m adlı alt program yukarıda verilmişti.

Yukarıda verilen $z = 5t^3 - 6t^2 + 8$ fonksiyonunun t=[-1:3] aralığındaki değişimi şekil 5.11'de görülmektedir. Bu çizimi yaptıran MATLAB komutu aşağıda gösterilmiştir:



Şekil 5.11



Şekil 5.12

Yukarıda kullanılan fplot (çizim) komutu ile ilgili tanımlar aşağıda verilmiştir;

fplot ('f.adı', 'limit', 'tol'): Bu komut MATLAB ortamında bir fonksiyonun çizimi için kullanılır. f. adı ile gösterilen yere ya MATLAB arşivinde yer alan hazır fonksiyonlar (ör:sin,cos,tan..) yada altprogram ile tanımlanan fonksiyonun yazıldığı dosyanın adı (ör: minbul.m) konur. Kullanıcı isterse iki tırnak arasına fonksiyonu da direkt olarak yazabilir. limit ile gösterilen yere ise [xmin xmax ymin ymax] değerleri yerleştirilir. Böylece çizimin hangi sınırlar arasında yapılacağı belirtilmiş olur. Eğer ymin ve ymax ifadeleri yazılmamış ise otomatik olarak MATLAB bu aralığı belirleyecek demektir. tol yazılan yerde sayı <1 ise buradaki sayının çizimin hata toleransı olduğu anlaşılır. Örnek olarak buraya 2e-3 yazılır ise hatanın %0.2 olacağı belirtilmiş olur. Eğer bu sayı ≥ 1 ise hesaba katılan nokta sayısı belirtilmiş olur.

Yukarıda verilen tüm işlemler aşağıda gösterilen şekilde de kolayca yapılabilir:

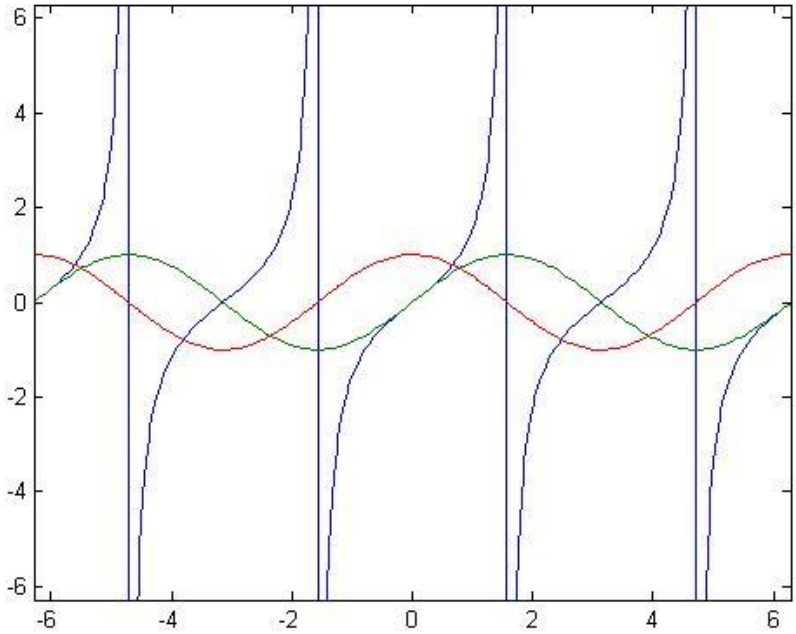
```
>> fminbnd('5*t.^3-6*t.^2+8',-1,3);
>> fplot('5*t.^3-6*t.^2+8',-1,3);
```

MATLAB ortamında tanımlı olmayan bir fonksiyonun (ör: l/sin fonksiyonu) x= [0.01:0.1] aralığında çizimini yapan program satırı aşağıda verilmiştir;

```
>> fplot(@(x) sin(1./x),[0.01 0.1],1e-3);
```

Yukarıda verilen program satırının uygulanmasından elde edilen çizim ise şekil 5.12'de gösterilmiştir, Yukarıdaki komut içinde kullanılan '@{x) ' ifadesi, değişkenin x olduğunu göstermektedir. Eğer birden fazla fonksiyon aynı eksen takımı üzerine fplot komutu yardımı ile çizdirilmek istenseydi aşağıdaki gibi bir komut kullanılabilirdi: (Çizim, şekil 5.13'de gösterilmiştir ve her bir eğri farklı renkli olarak çizilir)

```
>> fplot(@(x) [tan(x),sin(x),cos(x)],2*pi*[-1 1 -1 1]);
```



Şekil 5.13

Problem 5.15

$y = 0.025x^5 - 0.0625x^4 - 0.333x^3 + x^2$
fonksiyonunun minimum olduğu x değerlerini ve bu değerler için fonksiyonun aldığı y değerlerini x= [-1 4], x=[-4 -1], x= [-8 -4]aralıkları için ayrı ayrı bulan ve bu aralıklar için y(x) değişimlerini alt alta **aynı ekran** üzerine çizdiren MATLAB programı yazınız.

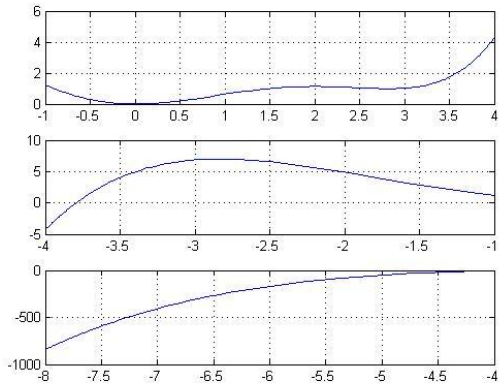
Çözüm

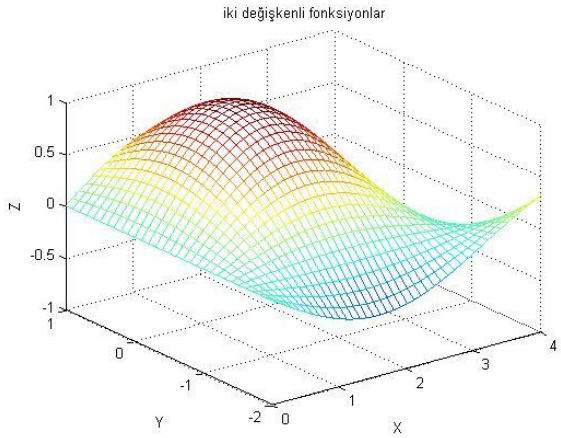
```
[xmin1 y1]=fminbnd('minbul1',-1,4)
subplot(3,1,1),fplot('minbul1',[-1 4]),grid
[xmin2 y2]=fminbnd('minbul1',-4,-1)
subplot(3,1,2),fplot('minbul1',[-4 -1]),grid
[xmin3 y3]=fminbnd('minbul1',-8,-4)
subplot(3,1,3),fplot('minbul1',[-8 -4]);grid

function y=minbul1(x)
y=0.025*x^5-0.0625*x.^4-0.333*x.^3+x.^2;
```

Yukarıda verilen programın çalıştırılması sonucunda elde edilen sonuç değerleri aşağıda verilmiştir. Elde edilen şekil ise şekil 5.14' de gösterilmiştir.

```
>>xmin1 =

  2.0438e-006

>> y1
```

```
y1 =

  4.1771e-012

>> xmin2

xmin2 =

   -1.0000
>> y2

y2 =

    1.2456

>> xmin3

xmin3 =

   -7.9999

>> y3

y3 =

 -840.6690
```





Şekil 5.14 Şekil 5.15

Problem 5.16

z =sin(x)*cos(y)
fonksiyonunu x ∈[0,4]ve y □ [- 2,l] aralığmda çizdiren MATLAB programı yazınız.

Çözüm

```
x=0:.1:4; y=-2:.1:1; [X,Y]=meshgrid(x,y); Z=sin(X).*cos(Y);
```

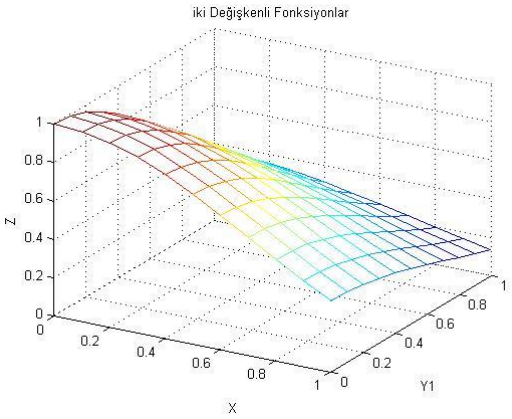
```
grid,mesh(X,Y,Z); title('iki değişkenli fonksiyonlar');
xlabel('X'), ylabel('Y'), zlabel('Z');
```

Yukarıda verilen MATLAB programının uygulanması sonunda elde edilen ekran görüntüsü şekil 5.15'de verilmiştir.

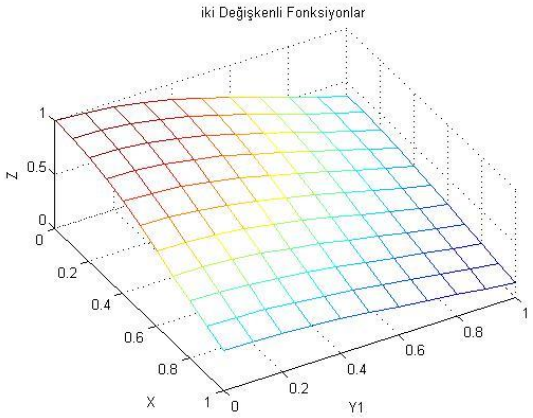
Problem 5.17

$f(x,y) = e^{-(x^2+y^2)}$ denklemi ile verilen üç boyutlu yüzeyi, x=[0;l],y=[0;l] aralığında çizdiren MATLAB programı yazınız.

Çözüm



Şekil 5.16



Şekil 5.17

Öncelikle verilen fonksiyon, yuzeyciz .m adlı bir function dosyasında tanımlanmalıdır;

```
function f=yuzeyciz(x,y)
f=exp(-x.^2-y.^2);
```

Verilen alanının çizimi için aşağıdaki MATLAB satırları incelenmelidir;

```
[X,Y]=meshgrid(0:0.1:1,0:0.1:1),
Z=yuzeyciz(X,Y),mesh(X,Y,Z),view(30,30),xlabel('X'),
ylabel('Y1'),zlabel('Z'),title('iki Değişkenli Fonksiyonlar');
```

Yukarıda verilen MATLAB satırlarının uygulanması sonunda elde edilen grafik şekil 5.16'da gösterilmiştir. Dikkat edilirse x ve y vektörleri meshgrid komutu içinde tanıtılabilmektedir. Program satırlarında kullanılan `view(60,60)`, elde edilen yüzeyin yatay ve düşey olarak kaç derece döndürüleceğini belirten bir MATLAB komutudur. Şekil 5.16'da view (60,60) yapıldığında (mevcut şekil, yatay ve düşey eksende 30° daha kaydırılıyor) şekil 5.16 yerine şekil 5.17'de gösterilen grafik elde edilir.

Yukarıdaki program aşağıdaki şekilde de yazılabilirdi:

```
[X,Y]=meshgrid(0:0.1:1,0:0.1:1),
Z=exp(-X.^2-Y.^2) ;mesh(X,Y,Z), view(30,30), xlabel('X'),
ylabel('Y'),zlabel('Z'),title('iki Değişkenli Fonksiyonlar');
```

BÖLÜM 6

İSTATİSTİKSEL ANALİZ

6.1. Maksimum ve minimum değerlerin bulunması

Minimum ve maksimum sayılarının bulunmasında MATLAB ortamında min ve max komutları kullanılır.

max (x) : Eğer x bir vektör ise bu komut ile x vektörünün en büyük elemanı bulunur. Eğer x bir matris ise bu komut ile x matrisindeki her bir sütun içerisinde bulunan en büyük sayı (bir vektör olarak) bulunur:

```
>> y=[-3    4   12;0 -2    9;4   5 6]

     y =
          -3     4    12
           0    -2     9
           4     5     6

>> max (y)

ans =
     4     5    12
>> b=[-1 2 0];
>> max(b)

ans =
      2
```

[a b] =max (c): Eğer c bir vektör ise bu komut ile c vektörünün **en büyük** elemanı bulunur ve bu a adlı değere atanır, bu değerin indisi ise b adlı değere atanır. Eğer cx bir matris ise bu komut ile c matrisindeki her bir sütun içerisinde bulunan en büyük sayı (bir vektör olarak) a vektörüne atanır, b ise a' yi oluşturan her bir elemanın içinde bulunduğu sütunun kaçıncı elemanı olduğunu gösteren vektördür.

```
>> c= [1   4   10;9   2 -8;0   5 -6]

c =
     1     4    10
     9     2    -8
     0     5    -6

>> [a,b]=max(c)

a =
     9     5    10

b =
     2     3     1
```

Eğer kullanıcı yukarıda verilen c matrisinin en büyük elemanım bulmak

isterse aşağıdaki işlemi yapabilir;

```
>> max(max(c))

ans =
     10
```

Eğer c matrisinin en büyük elemanının değeri ve bunun adresi bulunmak istenirse;

```
>> [m n]=max(max(c))

m =
    10

n =
     3
```

Yukarıdaki sonuca göre a matrisinin en büyük elemanı 10, bulunduğu sütun sayısı ise 3 olur.

max(a,b) : Bu komut ile a ve b matris elemanları birer birer karşılaştırılır, en büyük değer yeni oluşturulan matrise atanır. a ve b matrisleri aynı boyutta olmalıdır. Bu komut ile elde edilen matris de a ve b ile aynı boyuttadır.

```
>> a= [3 4 5;1 2 7]

a =
     3     4     5
     1     2     7

>> b= [0 1 2;10 -3 8]

b =
     0     1     2
    10    -3     8
>> max(a,b)

ans =
     3     4     5
    10     2     8
```

min (x) : Eğer x bir vektör ise bu komut ile x vektörünün **en küçük** elemanı bulunur. Eğer x bir matris ise bu komut ile x matrisindeki her bir sütun içerisinde bulunan en küçük sayı (bir vektör olarak) bulunur:

```
>> k= [10 9 2;40 18 2;3 5 2]

k =
    10     9     2
    40    18     2
     3     5     2

>> min(k)

ans =
```

```
     3     5     2

>> m= [0 1 2];
>> min(m)

ans =
     0
```

Eğer kullanıcı yukarıda verilen k matrisinin en küçük elemanını bulmak isterse aşağıdaki işlemi yapabilir;

```
>> min(min(k))

ans =
     2
```

[a b] = min (c) : Eğer c bir vektör ise bu komut ile c vektörünün en küçük elemanı bulunur ve bu a adlı değere atanır, bu değerin indisi ise b adlı değere atanır. Eğer c bir matris ise bu komut ile c matrisindeki her bir sütun içerisinde bulunan en küçük sayı (bir vektör olarak) a vektörüne atanır. b ise a'yı oluşturan her bir elemanın, içinde bulunduğu sütunun kaçıncı elemanı olduğunu gösteren vektördür:

```
>> c= [-1 8 3;0 -2 6;9 4 2]

c =
    -1     8     3
     0    -2     6
     9     4     2

>> [a,b]=min(c)

a =
    -1    -2     2

b =
     1     2     3
```

Eğer c matrisinin en küçük elemanının değeri ve bunun adresi bulunmak istenirse;

```
>> [m n]=min(min(c))

m =
    -2

n =

     2
```

Yukarıdaki sonuca göre a matrisinin en küçük elemanı -2, bulunduğu sütun sayısı ise 2 olur.

min (a,b) : Bu komut ile a ve b matris elemanları birer birer karşılaştırılır, en küçük değer, yeni oluşturulan matrise atanır. a ve b matrisleri aynı boyutta olmalıdır. Bu komut ile elde edilen matris de a ve b ile aynı boyuttadır.

```
>> a= [3 4 -8;10 5 9]

a =
     3     4    -8
    10     5     9

>> b= [1 4 7;-1 2 8]

b =
     1     4     7
    -1     2     8

>> min(a,b)

ans =
     1     4    -8
    -1     2     8
```

6.2. Vektör vematris elemanları arasında toplam ve çarpım işlemi

Eğer x, aşağıda tanımlandığı gibi N elemanlı bir **vektör** ise;

x=[x(l)x(2)x(3)…….. x(N)]

- x vektörünün tüm elemanlarının birbirleri ile toplanması sonunda elde edilen y bir **sayı** olur;

$$y=\sum_{n=1}^{N} x(n) = x(1) + x(2) + ... + x(N)$$

Yukarıda tanıtılan matematiksel işlemi MATLAB ortamında yapan komut aşağıda verilmiştir;

sum (x) : x bir vektör ise bu komut ile x vektörünün elemanlarının **toplamı** yapılır. Eğer x bir matris ise her bir sütunun toplamı ayrı bir eleman olarak atanır;

```
>> a= [-1 4 8]

a =
    -1     4     8

>> sum(a)

ans =
    11
```

```
>> m= [-1 12 32;2 15 -6;17 3 6]

m =
    -1    12    32
     2    15    -6
    17     3     6

>> sum(m)

ans =
    18    30    32
```

Eğer kullanıcı m matrisinin tüm elemanlarının toplamını bulmak isterse aşağıdaki işlemi yapmalıdır;

```
>> sum(sum(m))

ans =
    80
```

Problem 6.1

$\sum_{x=3}^{8} x$ toplamını hesaplayan MATLAB programını yazınız.

Çözüm

```
>> a= 3:8;
>> toplam=sum(a)

toplam =

    33
```

Yukarıdaki işlem aşağıda verilen komut satırını kullanarak da yapılabilir;

```
>> toplam=sum(3:8);
```

Eğer kullanıcı 3'den 15'e kadar 3'şer artışla elde edilen dizi elemanlarının toplamını elde etmek isterse, aşağıdaki komut satırını kullanabilir:

```
>> sum(3:3:15)

ans =
    45
```

x vektörünün tüm elemanlarının birbirleri ile çarpılması sonunda elde edilen y bir **sayı** olur;

$$y= \prod_{n=1}^{N} x(n) = x(1) * x(2) * \dots * x(N)$$

Yukarıda tanıtılan çarpım işlemini MATLAB ortamında yapan komut aşağıda verilmiştir;

factorial(a): Matematikte kullanılan **faktoriyel hesabı** için MATLAB ortamında factorial komutu kullanılır. Bu komut ile a bir vektör ise a vektörünün elemanlarının **çarpımı** yapılır. Eğer a bir matris ise her sütunun çarpımı ayrı bir eleman olarak atanır:

```
>> m= [10 6 4 3]

m =
    10     6     4     3

>> factorial(m)

ans =

     3628800         720          24           6
>> c= [1 3 2;10 20 23;2 11 3]

c =
     1     3     2
    10    20    23
     2    11     3

>> factorial(c)

ans =
  1.0e+022 *

    0.0000    0.0000    0.0000
    0.0000    0.0002    2.5852
    0.0000    0.0000    0.0000
   >> nfack=factorial(6)

   nfack =
      720
```

Aşağıda verilen komut 1 ile 3 (bu sayılar dahil) arasındaki tüm sayıların faktoriyel hesabını yapar:

```
>> factorial(1:3)


ans =


     1     2     6
```

Aşağıda verilen komut 2 ile 10 (bu sayılar dahil) arasındaki tüm sayıların 2'şer arttırarak elde edilen sayıların faktoriyel hesabını yapar:

```
>> factorial(2:2:10)

ans =
    2     24     720     40320     3628800
```

prod (a): **Çarpım hesabı** için MATLAB ortamında kullanılan diğer bir komut ise p r o d komutudur.

```
>> n=8;
>> ncarp=prod(1:n)

ncarp =
       40320
```

Yukarıdaki işlem aşağıda verilen komut satırını kullanarak da yapılabilir:

```
>> carp=prod(1:8)

carp =
       40320
```

Kullanıcı isterse 1 ile 10 arasındaki sayıların üçer artarak gitmesi durumunda elde edilen sayıların çarpımı (1*4*7*10-280);

```
>> carp1=prod(1:3:10)


carp1 =
   280
```

olarak hesaplanabilir.

Eğer kullanıcı 2'den 16'ya kadar 2'şer artışla elde edilen dizi elemanlarının çarpımım elde etmek isterse, aşağıdaki komut satırını kullanabilir:

```
>> prod(2:2:16)


ans =
    10321920
```

- Aşağıdaki işlem sonunda y bir **vektör** olur;

y=[y(1) y(2) y(3)……y(N)]

y'nin **kümülatif** toplamı;

$y(k) = \sum_{n=1}^{k} x(n) = x(1) + x(3) + x(2) + \ldots + x(k)$

y = [x(l) x(l)+x(2) x(1) + x(2) + x(3) .....]

Yukarıda tanıtılan matematiksel işlemi MATLAB ortamında yapan komut aşağıda verilmiştir;

c u m s u m (a); Eğer a bir vektör ise bu komut a ile aynı boyutta bir vektör oluşturur. Bu vektörün elemanları ise a'nın elemanlarının kümülatif toplamından meydana gelir. Eğer a bir matris ise bu komut a'nın sütun elemanlarının **kümülatif toplamını** yapar. Elde edilen matris yine a matrisi ile aynı boyuttadır.

```
>> a= [-4 7 12];
>> cumsum(a)
```

```
ans =
    -4     3    15

>> cumsum([-4 7 12])

ans =
    -4     3    15

>> k= [21 -4 16;12 -5 31;22 9 -2]

k =
    21    -4    16
    12    -5    31
    22     9    -2

>> cumsum(k)

ans =
    21    -4    16
    33    -9    4
    55     0    45
```

y'nin **kümülatif** çarpımı;

$y(k) = \prod_{n=1}^{k} x(n)$ = x(l)*x(2)*…..*x(k)

y = [x(l) x(l)*x(2) x(l)*x(2)*x(3) .....]

olur. Eğer x; M satırlı ve N sütunlu bir **matris** ise;

Yukarıda tanıtılan matematiksel işlemi MATLAB ortamında yapan komut aşağıda verilmiştir;

cumprod (a): Eğer a bir vektör ise bu komut a ile aynı boyutta bir vektör oluşturur. Bu vektörün elemanları ise a'nın elemanlarının kümülatif çarpımından meydana gelir. Eğer a bir matris ise bu komut a'nın sütun elemanlarının **kümülatif çarpımını** yapar. Elde edilen matris yine a matrisi ile aynıboyuttadır.

```
>> a= [-1 12 9 3];
>> cumprod(a)

ans =
    -1   -12  -108  -324

>> c= [22 14 9;10 8 7;12 5 2]

c =
    22    14     9
    10     8     7
    12     5     2

>> cumprod(c)

ans =
          22          14           9
         220         112          63
        2640         560         126
```

- Aşağıdaki işlem sonunda y bir **vektör** olur <u>(sütun toplamı)</u>.

$y(n) = \sum_{m=1}^{M} x(m,n) = x(1,n) + x(2,n) + \ldots + x(M,n)$

y = [x(l,l) + x(2,l) +.....+ x(M,l) x(l,2) + x(2,2) + ...+ x(M,2) ..... x(l,n) + x(2,n) + ...+ x(M,n)]

- Aşağıdaki işlem sonunda y bir **vektör** olur (sütun çarpımı).

$y(n) = \prod_{m=1}^{M} x(m,n)$ = x(l,n)*x(2,n)*...*x(M,n)

- Aşağıdaki işlem sonunda y; M satırlı, N sütunlu bir **matris** olur (kümülatif sütun toplamı).

$y(k,n) = \sum_{m=1}^{k} x(m,n)$ = x(l,n) + x(2,n) + ... + x(M,n)

- Aşağıdaki işlem sonunda y; M satırlı, N sütunlu bir **matris** olur (kümülatif sütun çarpımı).

$y(k,n) = \prod_{m=1}^{k} x(m,n) = x(1,n) * x(2,n) * \ldots * x(M,n)$

Problem 6.2

rand komutu ile K adlı 10*8 boyutunda bir matris üretilecektir. K matrisi satır satır taranacak, her satırın ortalaması A adlı vektöre, her satırın elemanlarının birbirleri ile toplamı B adlı vektöre, her satırın elemanlarının birbiri ile çarpımı ise C adlı vektöre atanacaktır. Bu işlemi yapan MATLAB programı yazınız.

Çözüm

```
>> K=rand(10,8);
>> A=mean(K');
>> B=sum(K');
>> C=prod(K');
```

Problem 6.3

Aşağıdaki işlemi MATLAB ortamında yapınız:

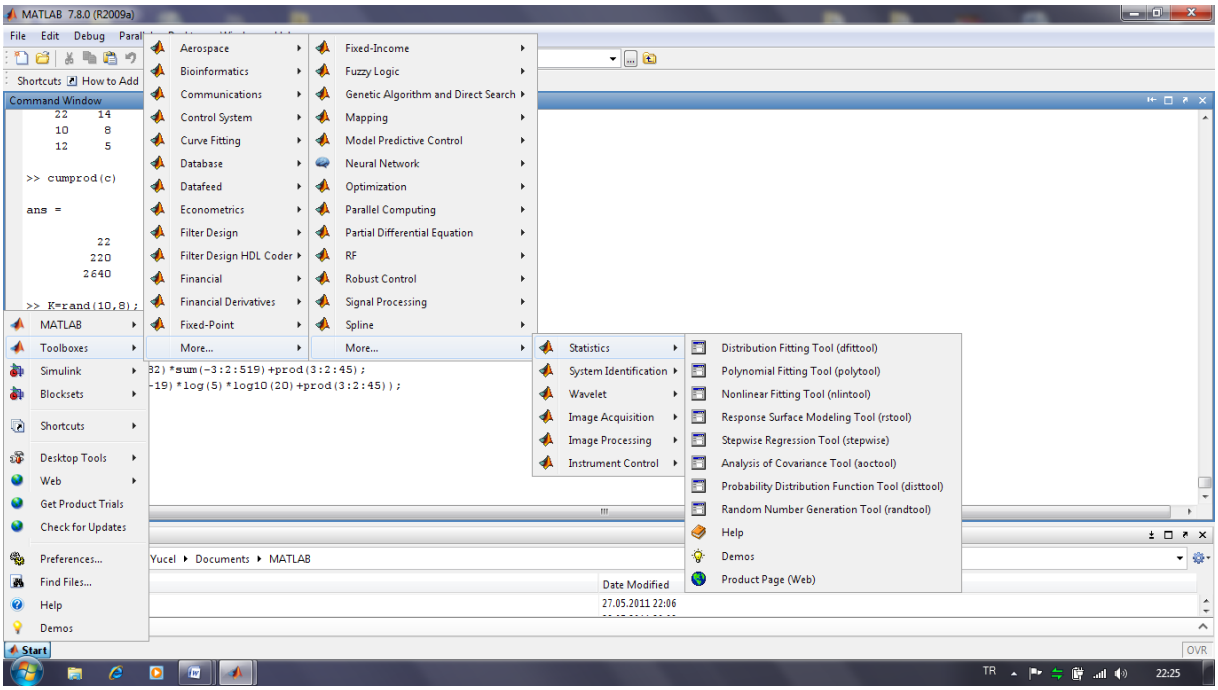
$$A = \frac{(784+787+790+793+.....+982)*(-3-1+1+3+5+7+.....+519)+(3*5*7*9.....*45)}{(-89-87-85-83-.....-19)*(\ln 5)*(\log 20)}$$

Çözüm

```
>> A1=sum(784:3:982)*sum(-3:2:519)+prod(3:2:45);
>> B1=(sum(-89:2:-19)*log(5)*log10(20)+prod(3:2:45));
>> sonuc=A1/B1
```

6.3. İstatistiksel analiz

MATLAB ortamında Statictics Toolbox içinde tanımlı olan komutlar yardımı istatistiksel analizler yapılabilir. Rastgele değişen bir data (veri) çeşitli şekillerde adlandırılabilir. Eğer data bir vektör ve aldığı değerler süreklilik gösteriyor ise, data **sürekli rastgele değişken** olarak adlandırılır. Eğer data, mümkün olan **tüm** değerlerin içinden rastgele olarak seçilmiş ise data bir **dağılımı** gösterir. Eğer sonlu sayıda data söz konusu ve bunların içinden örnek alınacak ise bu data grubu **örnek** olarak adlandırılır. Bazen data grubu içinde bazı değerlerin sıklık derecelerinin ölçümü de istenebilir. Bu nedenle ölçüm sonuçlarının **frekans** (sıklık) **dağılımı**'na ihtiyaç duyulur.



Şekil 6.1

Şekil 6.1'de MATLAB Startbutonu yardımı ile Statistics Toolbox menüsü içinden ulaşılabilen istatistiksel analiz Tool (araç) menülerine ulaşım yolu gösterilmiştir. Şekil 6.1'de MATLAB ortamında kullanıcıya 6 adet Tool seçeneği sunulduğu anlaşılmaktadır. Bu seçenekler DistributionFitting Tool, Polynominal Fitting Tool, Nonlinear Fitting Tool,Response Surface Modeling Tool, Stepwise Regression Tool, Analysis of Covariance Tool olarak verilmiştir.

6.3.1. Histogram

Histogram, ölçüm sonuçlarının dağılımını gösteren özel bir grafik çizimidir. Histogram, frekans dağılımını çubuk grafik olarak gösterir. Histogram eğrisinden elde edilen bilgi **ortalama** ve **varyans**'tan elde edilen bilgiden farklıdır. Histogram, değerlerin hangi aralıkta değiştiğini göstermekle kalmaz aynı zamanda onların bu aralıkta nasıl dağıldığını da (düzgün, Gaussian-normal,vb.) gösterir. MATLAB ortamında histogram eğrilerinin elde edilmesinde histkomutu kullanılır. Aşağıda bu komutun özellikleri tanıtılmıştır:

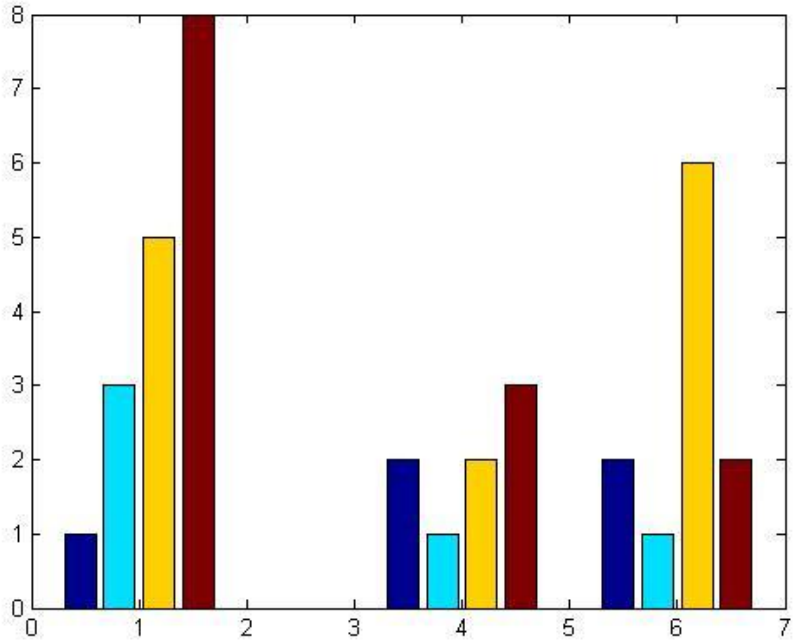
n=hist(x):	Eğer x bir vektör ise x'in alt ve üst limitleri arasında 10 adet kutu kullanarak histogram eğrisini çizer ve değerleri n isimli (10 elemanlı) vektöre atar. Eğer x bir matris ise n adındaki matrisin her sütunu 10 adet elemandan oluşur.
n=hist(x,m):	x'in alt ve üst limitleri arasında m adet kutu kullanarak histogram eğrisini çizer ve değerleri n adlı (m elemanlı) vektöre atar.
[n y]=hist(x,m}:	x'in alt ve üst limitleri arasında m adet kutu kullanarak histogram eğrisini çizer ve değerleri n adlı (m elemanlı) vektöre atar. y vektörü her bir kutunun (m adet) ortalamasını içerir. Bu komut genellikle çubuk (bar) grafiklerinin üretilmesinde kullanılır.

Yukarıda verilen komutlara ilişkin MATLAB uygulamaları **rastgele sayıüretimi** bölümünde verilecektir.

bar(x,y) : y matrisi m*n boyutundadır. Bu komut ile M adet çubuk grubu N farklı gruba ayrılır.Bu grupların yerleşeceği noktalar x vektörü ile belirlenir. Aşağıdaki örnekincelenmelidir:

```
>> a= [4 1 6];
>> b= [2 1 2 3;1 3 5 8;2 1 6 2];
>> bar(a,b);
```

Yukarıda verilen programın MATLAB ortamında uygulanması sonucunda elde edilen grafik şekil gösterilmiştir.



Şekil 6.2

bar(y) : y matrisi m*n boyutundadır. Bu komut ile M adet çubuk grubu N farklı gruba ayrılır.Bu grupların yerleşeceği noktalar x=1:M vektörü ile belirlenir. x vektörübelirtilmez ise x yerine x=1:M vektörü otomatik olarak atanır.

bar (y,x,genişlik}:genişlik komutu ile çubukların genişliği belirlenir, genişlik>1 içinçubukta binişim (üst üste geçme) olur. Eğer genişlik belirtilmez ise genislik= 0.8 alınır.

Bu komut ile ilgili uygulamalar, **rastgele sayıüretimi** konusunda verilecektir.

6.3.2. Aritmetik ve geometrik ortalama hesabı

mean (a) : a bir vektör ise bu komut ile a' nın elemanları toplanır ve eleman sayısına bölünür. Eğer a bir matris ise a' nın her bir sütununun ortalama değerini hesaplar:

```
>> b= [2 1 2 3;1 3 5 8;2 1 6 2]
b =
     2     1     2     3
```

```
     1     3     5     8
     2     1     6     2

>> mean(b)

ans =
    1.6667    1.6667    4.3333    4.3333

>> y= [1 2 3]

y =
     1     2     3

>> mean(y)

ans =
     2
```

geomean(x): x vektörünün geometrik ortalama değerini hesaplar.

$X=[X_1,X_2,X_3,\ldots X_n]$
vektörünün geometrik ortalaması;

$$\prod_{i=1}^{n}\frac{1}{n}$$

ifadesi ile hesaplanır. Aynı işlem MATLAB ortamında aşağıdaki gibi yapılır:

```
>> a= [2 4 3 7 4 2 1 8 2];
>> geomean(a)

ans =
    3.0296
```

harramean(x): x vektörünün harmonik ortalama değerini hesaplar.
$x=[x_1,x_2,x_3)\ldots x_n]$
vektörünün harmonik ortalaması;

$$\frac{n}{\prod_{i=1}^{n}\frac{1}{x_i}}$$

ifadesi ile hesaplanır. Aynı işlem MATLAB ortamında aşağıdaki gibi yapılır:

```
>> a= [1 2 3 2 4 6 4 8 12];
>> harmmean(a)

ans =
    2.8052
```

trimmean(x,k): x vektörünün (%k)/2' lik maksimum ve minimum değerlerinin atılması ile elde edilen yeni vektörün ortalamasını bulur.

```
>> a= [1 2 3 2 4 6 4 8 12];
>> trimmean(a,50)

ans =
    3.8000
```

Yukarıda a vektörünün %50 kırpılmış değeri hesaplanmaktadır.

median(x): x bir vektör ise x'in alt ve üst sının arasındaki orta değer (değerler sıralandığında ortada bulunan) elemanını bulur. Eğer x bir matris ise x'in her birsütununun içinde o sütundaki sayıların mean {x) değerini hesaplar;

```
>> y= [1 2 3];
>> median(y)

ans =
     2
>> a= [1 2 3; 2 4 6;4 8 12];
>> median(a)

ans =
     2     4     6
```

Yukarıda görüldüğü gibi b matrisinin ilk sütunu içindeki sayılar arasında 1 ile 4 arasında olup bu sütunun ortalama değerine en uygun olan sayı 2 olmaktadır. İkinci sütun için bu sayı 8, üçüncü sütun için ise 6 olarak görülmektedir.

median(x,k): x bir matris olmak üzere k=l ise x'in orta değeri x'in sütunları üzerinde hesaplanır. k=2 ise x'in her bir satın üzerinde orta değer hesaplanır;

sort(x): x bir vektör ise bu komut ile x'in elemanları küçükten büyüğe doğru sıralanır. x bir matris ise bu komut ile x'in sütunları küçükten büyüğe sıralanır;

```
>> a= [1 2 3; 2 4 6;4 8 12]

a =
     1     2     3
     2     4     6
     4     8    12

>> sort(a)

ans =
     1     2     3
     2     4     6
     4     8    12
```

sort(a,l): a matrisinin sütunlarını küçükten büyüğe sıralar.
sort(a,2): a matrisinin satırlarını küçükten büyüğe sıralar.
sort(a,'ascend'): a vektörünün elemanlarım küçükten büyüğe doğru sıralar.
sort(a,'descend') : a vektörünün elemanlarını büyükten küçüğe doğru sıralar.
sort(a,2,'descend') : a matrisinin satırlarını büyükten küçüğe sıralar.

6.3.3. İstatistiksel analizde kullanılan temel kavramlar

N elemanlı x vektörüne ilişkin standart sapma;

$$\sigma = \left[\frac{1}{N-1}\sum_{n=1}^{N}(x(n)-\bar{x})^2\right]^{0.5} \qquad (6.1)$$

ifadesi ile hesaplanır. Varyans değeri ise σ^2 olduğu unutulmamalıdır.

std(a): a bir vektör ise bu komut ile a vektöründeki değerlerin **standart sapması** hesaplanır. Eğer a bir matris ise bu komut a matrisinin her bir sütunu için standart sapma değerini hesaplar.

```
>> y= [13 2 9]

y =
    13     2     9

>> std(y)

ans =
    5.5678

>> var(y)

ans =
    31
```

Yukarıda da görüldüğü gibi var(a)=(std(a))^2 eşitliği sağlanmaktadır.

```
>> c= [1 2 3; 2 4 6;4 8 12]

c =

     1     2     3
     2     4     6
     4     8    12

>> std(c)

ans =
    1.5275    3.0551    4.5826

>> var(c)

ans =
    2.3333    9.3333   21.0000
```

cov(x) : x bir vektör ise bu komut ile x vektöründeki değerlerin **kovaryansı** hesaplanır.

```
>> a= [1 2 3 2 4 6 4 8 12];
>> cov(a)

ans =
   12.2500
```

corrcoef(x): x matrisinin her bir sütunu farklı veri setini içerir. Bu komut uygulandığında **korelasyonkatsayılarını** içeren matris elde edilir. Bu matrisin ana köşegen eleman değerleri her zaman1olur. Geri kalan elemanlar ise her bir sütunun diğer sütunlarda yer alan veri ile lineer ilişkisini gösterir. Elde edilen matris ana köşegene göre simetrik bir matris olur.

```
>> c= [8 2 5; 2 4 6;4 8 12];
>> corrcoef(c)

ans =
    1.0000   -0.5000   -0.3170
   -0.5000    1.0000    0.9799
```

```
   -0.3170    0.9799    1.0000
```

Örneğin x ve y adlı iki vektör arasındaki korelasyon incelendiğinde elde edilen matrisin (1,2) elemanı ile (2,1) elemanı daima eşit olur. Bu nedenle x ve y vektörleri arasındaki korelasyon elde edilen matrisin (1,2)veya (2,1) numaralı elemanına eşittir.

```
>> a= [1 2 3 -2];
>> b= [0.5 0.2 3 -0.4];
>> corrcoef(a,b)

ans =
    1.0000    0.7625
    0.7625    1.0000
```

Yukarıda görüldüğü gibi a ve b vektörleri arasındaki korelasyon 0.7391 dir.

```
>> a=1:0.2:5;
>> b=2*a;
>> corrcoef([a' b'])

ans =
      1      1
      1      1
```

Yukarıda elde edilen sonuçtan da görüldüğü gibi a ve b vektörleri birbirlerine tam olarak lineer bağlıdır.

6.3.4. **Düzgün dağılan rastgele sayılar**

Düzgün dağılan rastgele sayılar ya sabittir ya da minimum ve maksimum sayılar arasında düzgün olarak dağılan sayılardır, rand komutu [0 1]aralığında 'düzgün' olarak değişen sayılar üreten bir sayı üreticidir. Bu komut her kullanıldığında farklı sayılar elde edilir.

rand(n):	Bu komut ile değerleri [0 1] aralığında düzgün değişen n*n boyutunda bir matris oluşturulur.
rand(m,n) :	Bu komut ile değerleri [0 1] aralığında düzgün değişen m*n boyutunda bir matris oluşturulur.
rand :	Bu komut ile değeri [0 1] aralığında düzgün değişen bir sayı üretilir. Bu komut her yazıldığında elde edilen sayı farklı olur.
s=rand('state'):	Bu komut ile düzgün değişen sayı üreten üretecin o anda ürettiği 35 sayı, s vektörüne atanır.
rand('state',s):	Hafızadaki s vektörünü siler.
rand('state',0):	Üretecin başlangıç koşullarına döndürür.

Aşağıdaki örnekte rand komutu kullanılarak dataadlı data oluşturulmaktadır;

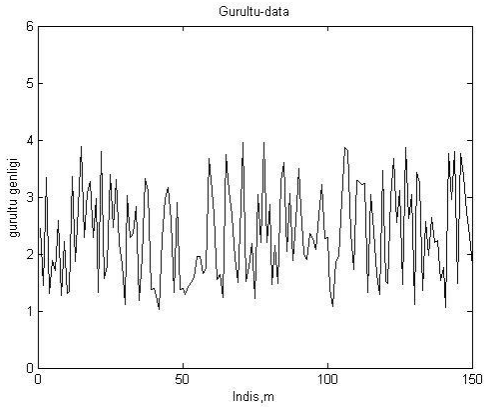
```
data=3*rand(1,150)+1;
save data;
plot(data,'k-');axis([0 150 0 6]);
title('Gurultu-data');xlabel('Indis,m');ylabel('gurultu
genligi');
```

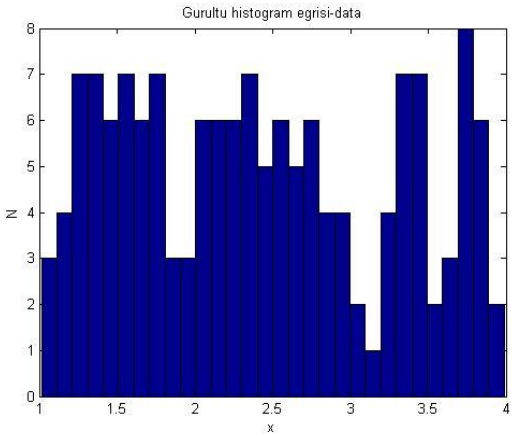
Yukarıda verilen MATLAB programında rand sayı generatörü (rastgele sayı üreten) tarafından üretilen sayılar 3 ile çarpılmakta ve elde edilen yeni sayı 1ile toplanmaktadır. Bu şekilde üretilen sayılar data adlı data dosyasında

saklanmakta ve sonra bu dosya değerleri şekil 6.3'de görüldüğü gibi çizdirilmektedir. Yukarıda üretilen `data` adlı data değerleri histogram eğri formunda da çizdirilebilir. Bu amaçla yazılan `MATLAB`programı aşağıda verilmiştir;

```
data=3*rand(1,150)+1;
hist(data,30);
title('Gurultu histogram egrisi-data');xlabel('x');ylabel('N');
```

rand komutu ile üretilen değerlerin `h i s t` komutu ile çizilen dağılımları şekil 6.4'te verilmiştir.





Şekil 6.3 Şekil 6.4

6.3.5. Normal (Gaussian) dağılan ratsgele sayılar

Bazı uygulamalarda elde edilen sonuçlar normal-Gaussian dağılımına uymaktadır. Normal dağılım fonksiyonunda iki önemli parametre kullanılır;

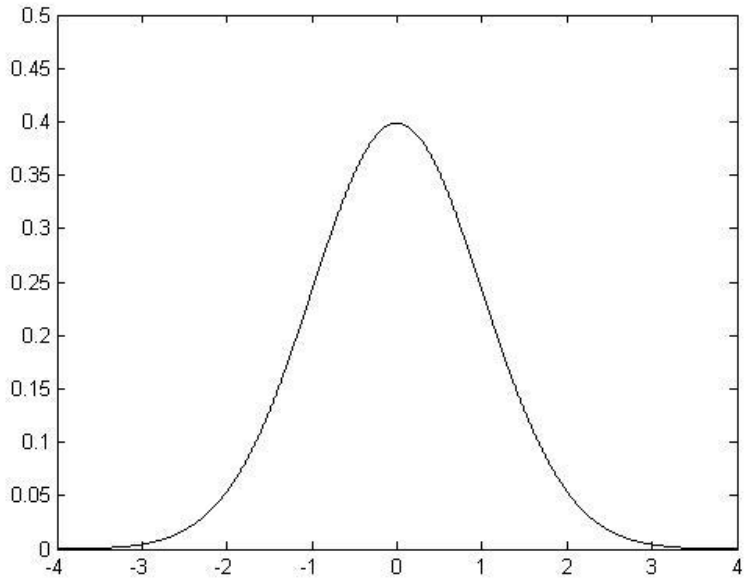
a) Dağılımın ortalama değeri (μ)
b) Dağılımın standart sapması (σ).

Normal dağılım;

$$f(x)= \frac{1}{\sqrt{2\pi\sigma}} e^{-(x-\mu)^2/2\sigma^2} \qquad (6.2)$$

fonksiyonu ile verilir. Şekil **6.5**'de $\mu=0$ ve $\sigma^2=1$ için normal dağılma gösterilmiştir. Bu eğriyi elde etmek için kullanılan MATLAB programı aşağıda verilmiştir;

```
x=-4:0.001:4; a=x.^2; c=1/sqrt(2*pi);f=c*((2.718).^-(a.*0.5));
plot(x,f,'k-'), axis([-4 4 0 0.5]);
```



Şekil 6.5

randn(n): MATLAB ortamında 'normal'-Gaussian dağılımı için randn komutu kullanılır, randn komutu 'normal' olarak değişen sayılar üreten bir sayı üretecidir. Bu komut her kullanıldığında farklı sayılar elde edilir. Bu komut ile ortalama değeri 0, standart sapması 1olan ve eleman değerleri 'normal' değişen n*n boyutunda bir matris oluşturulur.

randn(m,n): Bu komut ile ortalama değeri 0, standart sapması 1 olan ve eleman değerleri 'normal'değişen m*n boyutunda bir matris oluşturulur.

s=randn('state'): Bu komut ile 'normal' sayı üreten üretecin o anda ürettiği iki sayı s vektörüne atanır.

randn('state',s): Hafızadaki s vektörünü siler.

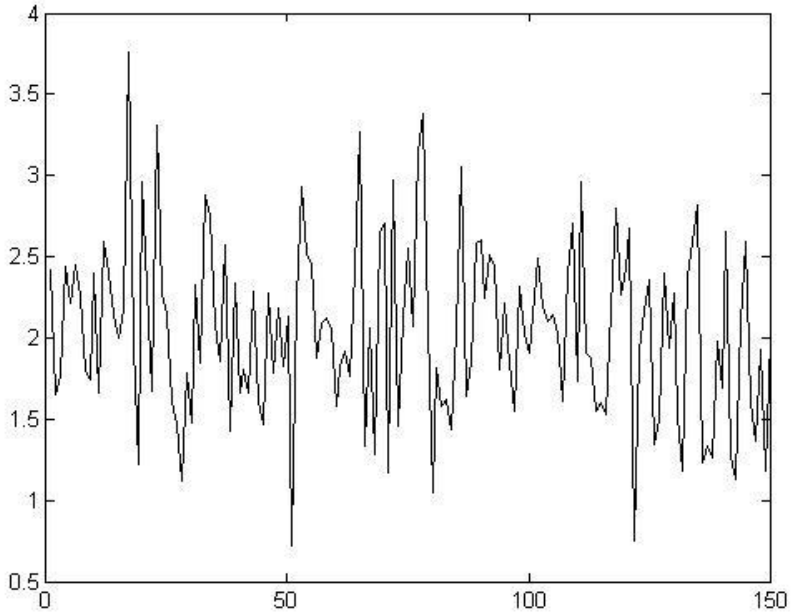
randn ('state',0): Üreteci başlangıç koşullarına döndürür.

Ortalama değeri **ort**, standart sapması **ss** olan (örneğin) 150 adet '**Gaussian**' rastgele sayısı içeren bir avektörü üretmek için;

```
>> a=ort+ss*randn(1,150)
```

MATLAB komutu kullanılabilir. Yukarıdaki a vektörünü üretip MATLAB ortamında çizdiren program aşağıda verilmiştir. Şekil 6.6'da ise a vektörünün değişimi çizdirilmiştir.

```
>> ort=2; ss=0.5;
>> a=ort+ss*randn(1,150);
>> plot(a,'k-');
```



Şekil 6.6

6.4.**Bozucu işaretinin**simülasyonu

Fiziksel büyüklükleri elektriksel işaretlere dönüştüren çoğu algılayıcılar, bozucu işareti olarak adlandırılan 'ölçüm hataları' üretirler. Gaussian sayı üreteci yardımı ile bu hatanın modellenmesi mümkün olabilir. İşaret modeli;

$x = s + n$ (6.3)

olarak verilebilir. (6.3) ifadesinde s; ilgilenilen gürültüsüz işareti, n; Gaussian ölçüm hatasını, x; gürültülü işareti (bozucu) göstermektedir. İşaret büyüklüğünün ölçümü **işaret-gürültü oranı** (IGO) olarak adlandırılır ve;

$$IGO = 10 \log_{10} \frac{\sigma_s^2}{\sigma_n^2} = 20 \log_{10} \frac{\sigma_s}{\sigma_n} \quad (6.4)$$

ifadesi ile hesaplanır. (6,4) eşitliğinde σ_s; işaretin standart sapması, $\boldsymbol{\sigma_n}$; gürültünün standart sapması olarak kullanılmıştır.

İçinde gürültü işareti (Gaussian formunda) barındıran sinüs formunda değişen bir işaretin simülasyonu yapılsın. Sinüs dalgasının varyansı;

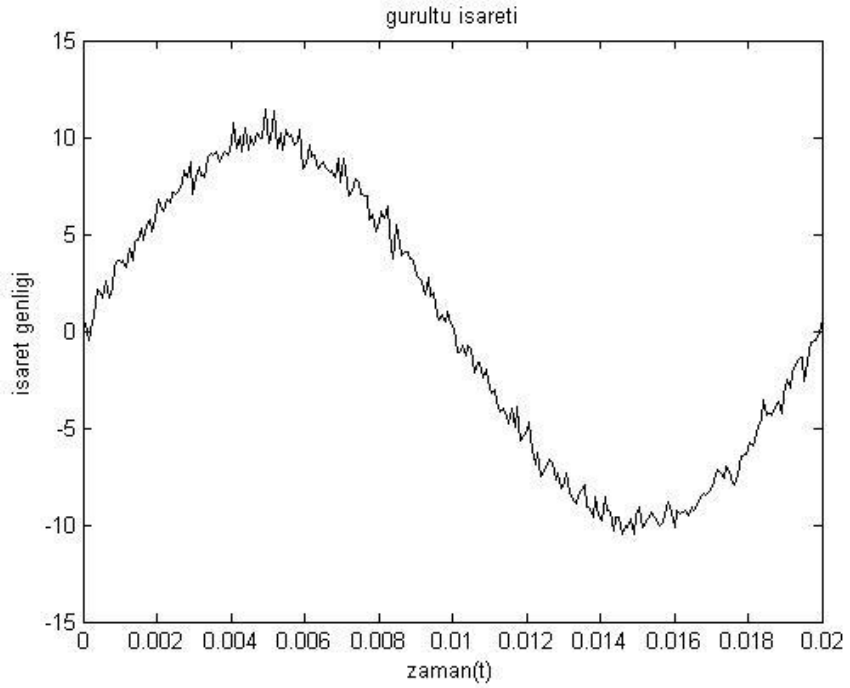
$$\sigma_s^2 = \frac{A}{2} \quad (6.5)$$

olur. (6.5)ifadesinde A; sinüs işaretinin genliğini göstermektedir. Aşağıda verilen MATLAB yazılımı (editör ortamında), aşağıda tanımı verilen bozucu (gürültü) işaretinin simülasyonunu yapmaktadır;

```
t=linspace(0,20*10^-3,256);
a=10*sin(2*pi/(20*10^-3)*t);
d=0.5*randn(size(t));
gurultu=a+d;
disp('isaret-gurultu orani(IGO),dB');
IGO=20*log10(std(a)/std(d));
plot(t,gurultu,'k-');
```

```
xlabel('zaman(t)');
ylabel('isaret genligi');title('gurultu isareti');
```

Yukarıda elde edilen IGO değeri ise `23.6799` dB olmaktadır. Bu sonuç (6.4) eşitliği ile hesaplanan değere çok yakındır. Aradaki fark ise tahminde kullanılan örnek sayıların sonlu sayıda olmasından kaynaklanmaktadır. Şekil 6.7'de yukarıda yazılan programdan elde edilen çizim gösterilmiştir.



Şekil 6.7

BÖLÜM 7

MANTIK FONKSİYONLARI

7.1. Mantık işlemleri

Mantık yada ilişki içeren ifadelerde; eğer giriş verileri sıfırdan farklı ise çıkış değerleri doğru (1), giriş verileri sıfır İse çıkış değerleri yanlış (0) olarak alınır. Bu anlamda çıkış değerleri mantıksal

(Iojik) değerler (1 ve 0 gibi) almaktadır. MATLAB ortamında kullanılan karşılaştırma işaretleri Tablo 7.1'de verilmiştir;

Tablo 7.1

işaret	Anlamı
<	Daha .küçük
<=	Daha küçük veya eşit
>	Daha büyük
> =	Daha büyük veya eşit
= =	Eşit
~ =	Eşit değil

Aşağıdaki MATLAB yazılımları incelenmelidir:

```
»A=l:9                                  ('enter')
A=
    1  2  3  4  5  6  7  8  9
»B=8-A                                  ('enter')
B =
     7  6  5  4  3  2  1  0  -  1
% Asagidaki satirda A vektörü içinde 4'e esit ve bundan kucuk olan sayılar l'e, diğerleri ise O'a
eşitleniyor
»secl= A<=4                             ('enter')
secl=
    1  1  1  1  0  0  0  0  0
% Asagidaki satirda A vektörü içinde B den buyuk olan sayılar l'e diğerleri ise O'a eşitleniyor
»sec2= A>B                              ('enter')
sec2=
0   0   0   0   1   1   1   1   1
%   A vektör içinde B ye esit olan sayilar l'e diğerleri ise O'a
eşitleniyor
»sec3= (A= =B)                          ('enter')
sec3 =
      0   0   0   1   0   0   0   0   0
»C=A>2    % 2'den büyük sayılar l'e, 2'ye esit ve 2'den kucuk sayilar ise O'a eşitleniyor
C =
       0     0     1     1     1     1     1     1     1
% Asagidaki satirda A nın içinde 2 den buyuk olmayan sayılar sıfıra eşitleniyor, D adlı vektöre
ataniyor
»D=C. *A
 D =
       0     0     3     4     5     6     7     8     9
```

Yukarıda verilen işlemlerde görüldüğü gibi '=' işareti ile '= =' işareti arasında fark vardır. '= =' İşaretiiki
değişkeni **karşılaştırır.** Eğer iki değişken aynı değeri alır ise sonuç '1', farklı ise sonuç '0' olur. "

Diğer taraftan '=' işareti ise 'atama' için kullanılır. '=' işaretinin sağ tarafındaki sayı bu işaretin sol tarafındaki değişkene **atanır.**

7.2. Sıfıra bölmeden kaçınma

MATLAB ortamında bazen sıfıra bölme işlemi ile karşılaşılır. Bu durumda MATLAB bu işlemi ifade etmek için NaN (Not **a** Number) komutu kullanır. Aşağıda verilen MATLAB yazılımı incelenmelidir;

```
» x = (-2:2)/2                          ('enter')
x =
     -1.0000   -0.5000         0    0.5000    1.0000
» sin(x) ./x                            ('enter')
Warning: Divide by zero. <- (uyrıda sıfıra bölme hatası olduğu belirtiliyor)
ans =
      0.8415   0.9589.          NaN   0.9589   0.8415
```

x vektörünün 4. elemanı sıfır olduğu için sin (x) /x ifadesinde yerine konulduğunda sin(0)/0 elde edilir. Bu ise 0/0 tanımsızlığmı ortaya çıkarır. Böyle bir işlem MATLAB ortamında NaN komutu ile ifade edilir. Yukarıdaki hesaplamada sıfır sayısından kurtulmak için MATLAB ortamında özel bir sayı olan eps değeri

sıfır yerine kullanılır, eps sayısı MATLAB'da $2.2*10^{-16}$ değerine eşit olan bir sayıdır. Bu sayının kullanılması ile yukarıda verilen hesaplamalar aşağıdaki şekli alır ve böylece sıfıra bölüm hatasından uzaklaşılmış olur;

```
» x = x + (x= =0)*eps                          ('enter')
x =
    -1.0000   -0.5000   O.OOOO   0.5000    1.0000
>>sin(x) ./x                                   ('enter')
ans =
  0.8415   0.9589    1.0000    0.9589    0.8415
```

Yukarıda verilen MATLAB satırında x=x+ (x==0) *eps ifadesi **şu** şekilde işlemektedir; Eğer x değeri sıfıra

Eşit ise (x==0) ifadesinin değeri 1 olmaktadır.'(x==0) *eps ' değeri bu durumda $l*eps=l*2.2*10^{-16}$

$= 2.2 *10^{-16}$ değerine eşit olmaktadır. Böylece x değeri sıfır olmadan, sıfıra yakın bir değere atanarak sıfıra bölüm hatasından kaçınılır.

7.3.1. Mantıksal işlemciler

Tablo 7.2'de MATLAB ortamında kullanılan mantıksal işlemcilerin lojik karşılıkları

Tablo

İşaret	Lojik karşılığı
&	ve
I	veya
~	değil

gösterilmiştir.
MATLAB ortamında kullanılan dördüncü mantıksal işlemci xor komutudur;

xor (A, B) : A ve B'nin her ikisi doğru veya her ikisi yanlış ise 0 değerini, bunun dışındaki seçeneklerde ise 1 değerini alır.

Tablo 7.3'te 4 adet mantıksal operatörün değişik durumlarda aldığı değerler gösterilmiştir.

Tablo 7.3

A	B	~A	A I B	A&B	xor(A,B)
0	0	1	0	0	0
0	1	1	1	0	1
1	0	0	1	0	1
1	1	0	1	1	0

Aşağıdaki MATLAB yazılımları incelenmelidir:

```
» A=l:9                                   ('enter')
A =
      1    2    3    4    5    6  .  7  8    9    '
% A içinde 4 den buyuk sayilar l'e diğerleri O'a eşitleniyor
» mant l = A>4                            ('enter')
mantl =
       0    0    0    0    1    1    1    1    1    -
% A içinde 4 den buyuk olmayan sayilar 1'e,diğerleri O'a eşitleniyor
» mant2 = ~(A>4)                          ('enter')
mant2 =
       1    1    1    1    0    0    0    0    0
% A içinde 2 den buyuk ve 6 dan kucuk sayilar l'e diğerleri O'a % eşitleniyor
» mant3 = (A>2)&{A<6)                     ('enter')
mant3 =
       0    0    1    1    1    0    0    0    0
% A içinde 2 den buyuk olmayan ve 6 dan kucuk olmayan sayilar

» mant4 = xor((A>2), (A<6))       ('enter')
 mant4 =
1    1    0    0    0    1    1    1
```

$$x(t)=\begin{cases}\sin(t) & \sin(t)>0\\ 0 & \sin(t)<0\end{cases}$$

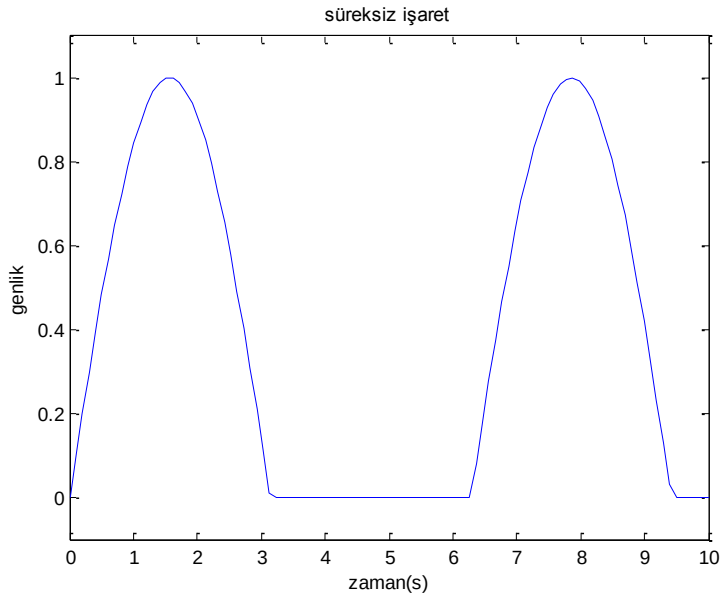
Örnek olarak x(t) süreksiz işareti;

ifadesi ile verilsin. [0 10] saniye aralığında bu işaret mantıksal işlemciler kullanılarak üretilsin ve çizdirilsin:

```
t=linspace(0,10,100);
```

```
x=sin(t);
x=x.*(x>0);        % en önemli program satiri
plot{t,x),;xlabel('zaman(s)'),ylabel('genlik'),title('süreksiz işaret'};
axis{[0   10   -0.1   1.1]);
```

Yukarıda verilen MATLAB programı sonucunda elde edilen değişimin çizimi şekil 7.1 'de gösterilmiştir. Programda 'x =x.*(x>0) ' satın iki vektörün nokta çarpımından oluşmaktadır. (x>0) vektörü 1 ve O'lardan oluşur. Bu vektörde, x'in elemanları içinde sıfırdan büyük olan sayılar l'e, diğerleri O'a atanır. Bu vektör ile x vektörü noktasal çarpılırsa (x.*{x>0)) ortaya çıkacak vektör x boyutunda, fakat pozitif olmayan elemanları 0 değerinde olan bir vektör olacaktır. Ortaya çıkacak yeni vektörün diğer elemanları x'in pozitif elemanİan ile aynı değerde olacaktır. Bu aralık çizim üzerinde yatay olarak görülen bölgeye karşı gelmektedir.



Şekil 7.1

MATLAB ortamında, çeşitli amaçla kullanılabilecek mantık fonksiyonları tanımlanmıştır. Aşağıda bu fonksiyonların özellikleri verilmiştir;

any (x) : x vektörü içinde herhangi bir eleman sıfırdan farklı ise bu fonksiyon 1 değerini almakta, aksi halde 0 değerini almaktadır. Eğer x bir matris ise x'in her bir sütunu incelenmekte,
oluşturulan vektör elemanları; eğer incelenen sütun içinde herhangi bir sayı sıfırdan farklı ise 1 aksi halde 0 değerini almaktadır.

```
» x=[ 3 -1   0   4];                (‘enter')
»any (x)                            ('enter')
ans=
1
» b=[0     1       3     -1;          ('enter')
     0       2      0      1;
0       1       0       3;
0     -  4  2     0 ];
```

»any(b) ('enter')

ans =
0 1 1 1

all (x) : x vektörü içindeki **tüm elemanlar** sıfırdan farklı ise bu fonksiyon 1 değerini almakta, aksi halde 0 değerini almaktadır. Eğer x bir matris ise x'in her bir sütunu incelenmekte,
oluşturulan vektör elemanları; eğer incelenen sütun içinde tüm elemanlar sıfırdan farklı ise
1 aksi halde 0 değerini almaktadır.

» x=[3 -1 0 4]; ('enter')
>>all(x) ('enter')
ans=
0
» b=[0 1 3 -1; ('enter')
0 2 0 1;
0 1 0 3;
0 - 4 2 0];
»all(b)
('enter')
ans=
o 1 0 o

7.3.2. Mantıksal kontrol **işlemcileri**

Tablo 7.4'de gösterilen mantıksal kontrol işlemcileri, her iki yanma yerleştirilen ve (1*1) boyutunda olan ifadenin doğru veya yanlış olmasına göre 1 veya 0 üreten MATLAB komutlandır.

Tablo 7.4

işaret	Lojik karşılığı
&&	ve
| |	veya

Kontrol işareti && ise ve bu işaretin her iki tarafındaki (1*1) boyutundaki ifadeler doğru ise sonuç 1, aksi halde 0 olacaktır. Eğer | | işaretinin her iki tarafındaki ifadelerden en az biri doğru ise sonuç 1, aksi durumlarda sonuç 0 olacaktır. Aşağıdaki örnekler incelenmelidir:

»a=2,b=3; ('enter')
>>(a<3)&&(b>4) ('enter')
ans =
0
»m=[l 2]; ('enter')
»n=[3 4] ; ('enter')

>>(m<3)&& (n>2) ('enter')
??? Operands to the | | and && operators must be convertible to logical scalar values.

Yukarıda görüldüğü gibi m ve n vektörleri (1*1) boyutunda olmadığı için hata uyarısı ile karşılaşılmıştır.

```
>>a=[1 0 -1 6]

a=

1   0  -1  6

» find(a)

ans =

1      3      4          » b=[0

1   3  -1;
0    1    0    3;
0   2   0  1;
0   - 4 2   0];
» find(b)
 ans =

5 6 7 8 9

12 13 14 15
```

find komutunun yanındaki ifade **lojik** bir fonksiyon ise bu komutun karşılaştırma amaçlı kullanılması da mümkündür. Aşağıdaki örnekler incelenmelidir:

```
»[satir, sütun] =find(b==2)      ('enter')
satir =
        3     %b matrisinde 2'ye esit ilk elemanİn satir numarasi
        4     %b matrisinde 2'ye esit ikinci elemanin satir numarasi
sütun =
        2     %b matrisinde 2'ye esit ilk elemanin surun numarasi
        3     %b matrisinde 2'ye esit ikinci elemanin sütun numarasi
```

Yukarıdaki program satırında b matrisinin 2'ye eşit elemanlarının adresleri aranmaktadır. Komutun uygulanmasından elde edilen sonuçlara bakıldığında b matrisinde iki tane 2 sayısı bulunduğu anlaşılmaktadır. İlk 2 sayısı 3. satır 2. sütunda, ikinci 2 sayısı ise 4. satır 3. sütunda bulunmaktadır.

Eğer b matrisinde 2 'den büyük elemanların adresleri merak edilirse, aşağıdaki komut satın uygulanmalıdır:

```
» [satir,sütun]=find(b>2)       ('enter')
```

```
satir =
        1
        2
sütun =
        3
        4
```

Elde edilen sonuçlara bakarak b matrisinde **1.** satır **3.** sütunda ve 2. satır 4. sünınunda 2'den büyük elemanlar bulunmaktadır. Eğer kullanıcı bu eleman değerlerini merak ederse aşağıdaki komut satırlarım kullanabüir:

```
» b(l,3)                                ('enter')
ans =
          3 .
» b(2,4)                                ('enter')
ans .=
          3
```

Eğer kullanıcı b matrisi içinde -5'den büyük, -l'den küçük eleman (yada elemanlann) adreslerini merak ederse aşağıdaki komut satirini kullanabilir:

```
» [satir,sutun]=find(b>-5 & b<-l)                ('enter')
satir = 4
sütun = 2
```

Elde edilen sonuca göre yukarıdaki şartı sağlayan bir adet eleman bulunmaktadır ve bunun adresi ise 4. satır 2. sütundur.

Aşağıda verilen örnekte ise b matrisinin 2 'den büyük elemanları 6 yapılmaktadır;

```
>>b(find(b>2) )=6                                ('enter')
b=
  0     1     6   -1
  0     1     0     6
  0     2     0     1
  0    -4     2     0
```

Aşağıdaki örnekte ise b'nin 1 'den büyük ve 3*ten küçük elemanları 8 yapılmaktadır:

```
» b(find(b>l & b<3))=8                          ('enter')
b=
  0     1     3   -1
  0     1     0     3
  0     8     0     1
  0    -4     8     0
```

Problem 7.1

Şekil 7.2'de, [-8;8] aralığında yl(t) eğrisinin y2(t) eğrisinden daha büyük veya eşit olduğu zaman aralığının alt (talt) ve üst sınırını (tüst) find komutu ile bulduran bir program yazınız.

Çözüm ■

```
»t=-8:0.01:8;
>>yl=-(t.^2-16);    y2=t.^2;
»u=f ind(yl>=y2) ;
»hold on; plot{t,yl); plot(t,y2) % eğriler cizdiriliyor
»talt=t(u(1))
»tust=t(u(end));
talt=
    -2.820
Tust=
     2.820
```

Problem 7.2

Kullanıcı, klavye yardımı ile bir elemanları reel sayılar olan bir matris girecektir. Yazılan MATLAB programı ile bu matristeki pozitif sayılar bir matrise, negatif sayılar bir diğer matrise ve 1 'den küçük veya 5'den büyük sayılar ise başka bir matrise atanacaktır. Bu işlemi yapan bir MATLAB programı yazınız.

Çözüm

```
A=input('[…;….;..]seklinde bir matris giriniz=')
k=find(imag(A));        %matristeki komplex sayıların yeri bulunuyor
b=A(k);                 %komplex sayıların yeri b ye atanıyor
disp('simdi   ekranda matrisin sadece negatif sayıları gözüksün')
A1={A<0).*A
dispCsimdi de ekranda sadece matrisin pozitif sayıları gözüksün')
A2=(A>0).*A
dispC simdi ise A matrisinde <1 veya >5 olan sayilari görelim')
A3=(A<1|A>5).*A
```

Yukanda verilen programın çalıştırılması sonunda elde edilen Command Window ekran görüntüsü aşağıda verilmiştir:

```
[ ---- ;------;…] seklinde bir matris giriniz=[l 2 3;456;3 0 -5]
A =
1     2     3
4     5     6
3     0    -5
simdi   ekranda matrisin sadece negatif sayıları gözüksün
Al =
0     0     0
0     0     0
```

0 0 -5

simdi de ekranda sadece matrisin pozitif sayıları gözüksün

A2 =

1 2 3
4 5 6
3 0 0

simdi ise A matrisinde <1 veya >5 olan sayilari görelim

A3 =

0 0 0
0 0 6
0 0 - 5

Problem 7.3

rand komutu ile K adlı 10*8 boyutunda bir matris üretiniz, rand komutu ile üretilen bu matrisin 0.2 den küçük elemanlarının değerleri ve adreslerini veren M adlı bir matris oluşturulacaktır. M matrisinin ilk sütunu; 0.2'den küçük sayılarının K matrisinin hangi satınnda bulunduğunu, M matrisinin ikinci sütunu; 0.2'den küçük sayınların K matrisinin hangi sütununda bulunduğunu ve M matrisinim üçüncü sütunu ise 0.2'den küçük sayınlann değerini verecektir. Bu işlem yapan MATLAB programını yazınız.

Çözüm

```
K=rand(10,8)
[a b]=find(K<0.2); % a satır numarası, b sütun numarası
 ss=length{a) ,*  % a nın satır sayısı
    for u=l:ss
   c(u,l)=K(a(u,l),b(u,l)); % aveb adreslerindeki K eleman değerlerini verir
    end
M=[a b c] % arzu edilen sonuç matrisi;
```

isnan(x) : x elemanları NaN ise bu komut ile oluşturulan vektörün elemanlarının değeri 1, bunun dışında ise 0 olarak atanır.Eğer x bir matris ise x'in eleman değeri NaN olduğunda oluşturulacak matrisin elemanlan *1*, aksi halde 0 olur.

» b=[-l 0,9589 NaN 2 0]; isnan(b)

('enter') (‘enter’)

0 0 1 0 0

```
>>C=[0  1   3    -1;
0          2  NaN   1;
      -1 NaN  1    -3;
       0   -4   2    0];
```

»isnan(c}
ans=
7 1 0

isfinite(x) :	x vektörünün elemanları sonlu bir sayı ise oluşan vektörün eleman değeri 1, aksi halde 0 olur. Sonsuz bir sayı MATLAB ortamında inf ile gösterilir.

isinf(x) :	x elemanları -inf veya inf ise bu komut ile oluşan vektörün elemanları 1, aksi halde 0 olur.

isempty(x)s	Eğer x matrisi boş ise bu komut 1, aksi halde 0 üretir.

isequal(a,b):	a ve b matrislerinin birbirlerine eşit olup olmadığmı kontrol eder. Eğer bu iki matris birbirlerine eşit ise sonuç 1 aksi halde sonuç 0 olarak elde edilir.

isreal(a) :	a büyüklüğü (vektör veya matris de olabilir) içinde tüm sayılar reel (gerçek sayı) ise 1 aksi halde 0 üreten bir komuttur.

islogical(b):	b büyüklüğü (dizi-matris yada vektör) hep lojik sayılardan oluşuyor ise 1, aksi halde 0 üreten bir komuttur.

Problem 7.4

Hacmi, yüzeyi ve ağırlığı ihmal edilebilen bir cisim yere monte edilmiş bir firlatıcı tarafından v_0 =20m/s

ilk hızı ile yatay eksenle 0 = 40° yapacak şekilde fırlatılmaktadır. Yerçekimi ivmesi g=9.81 m / sn^2 olduğuna göre, hem cismin hızının 16 m/s'ye eşit ve bu değerden küçük olduğu, hem de cismin çıktığı yüksekliğin 6 m'den büyük olduğu zaman aralığım bulunuz.

Çözüm

Cismin uçuş uzaklığı;

h(t) = vot*sinQ-0.5*g*t2

ve cismin havadaki hızı;

$$v(t) = \sqrt{Vo^2 - 2Vog * t * sin\theta + g^2 * t^2}$$

ifadeleri ile hesaplanır. Problemde sorulan zaman aralığı h(t) ve v(t) eğrilerinin çizimi yapılarak ta bulunabilir fakat bu işlem zaman alır. Problemin çözümünü (editör ortamında) yapan MATLAB yazılımı aşağıda verilmiştir:

```
% sabit değerleri gir
vO = 20;g=9.81;teta = 40*pi/180;
% ucus suresini hesapla
```

```
tu=2*v0*sin(teta}/g;
% zaman ileilgili olcum hassasiyetini belirle
t=[0:tu/200:tu];
h=vO*t*sin(teta)-0.5*g*t.^2;
v=sqrt(vO^2-2*v0*g*sin(teta)*t+g^2*t.^2);
% yükseklik 6 metre den az olmasın
%hiz 16 m/s den cok olmasin
u = find(h>6 £ v <= 16);
% incelenen zaman araliginin baslangic ve son değerini hesapla tbas = t(u(l)) tson = t(u(end))
subplot{2,l,l),plot(t,v), xlabel{[1] zaman' ) , ylabel(*hiz*);
subplot<2,l,2),plot(t,h), xlabel{[1] zaman' ) , ylabel{'yükseklik');
```

Yukarıda verilen yazılımın sonunda elde edilen zaman değerleri;.

»

```
tbas =
        0.8518
tson =
        1.7691'
```

olmaktadır. Yazılım sonunda (elde edilen değerleri kontrol etmek için) değişimler aynı zamanda çizdirilmİştir. Şekil 7.4'de verilen hız değişim eğrisinde görüldüğü gibi hızın 16 m/s'den küçük olduğu aralık tbas ve tson zaman aralığıdır. Yüksekliğin zamanla değişiminin gösterildiği ikinci grafikte ise yüksekliğin 6 m'den düşük olduğu zaman aralıkları A ve B noktaları arasında kalmaktadır.

Yazılımda belirtildiği gibi her iki zaman aralığının da sağlandığı (zira ve-&- mantıksal işlemcisi kullamlmaktadır) aralık tbas ve tson aralığıdır. Şekil 7.4 ile program sonunda elde edilen sonuçların örtüştüğü görülmektedir. Yazılımda tbas = t(u(l)) ifadesi ile u vektörünün ilk elemanı tbas'a atanırken, tson=t(u(end)) ifadesi ile de u vektörünün en son elemanı tson'a atanmaktadır.

7.5. Basit if bildirimi

MATLAB ortamında kontrol döngüsü olarak adlandırılan komutlardan bir tanesi de basit if bildirimidir. Bu bildirim yapısının MATLAB ortamında kullanılışı aşağıda gösterilmiştir:

```
if   'mantıksal deyim"
       'bildirim grubu'
end
```

Yukanda gösterilen bildirimde mantıksal deyim doğru ise (lojik olarak 1), 'bildirim grubu'nda yazılan işlemler icra edilir (uygulanır). Eğer 'mantıksal deyim' yanlış ise (lojik olarak 0), 'bildirim grubunda' yazılan işlemler uygulanmaz ve end komutunun altına geçilir. Aşağıdaki örnek incelenmelidir:

```
if  b>45
      toplam=toplam+l;
end
```

Yukrıda verilen programda b sayısı 45'ten büyük olduğunda 'toplam' değişkeni bir artırılır. Eğer 'toplam' değişkeni 45 sayışma eşit veya 45'ten küçük ise toplam değişkeni değer değiştirmemekte, diğer bir ifade ile toplam=toplam+l satırı uygulanmamakta ve end komutunun altına geçilmektedir.

Yukarıda verilen programda toplam=toplam+l satınnın if ve end komutları ile aynı hizada yazılması şart değildir. Bu tür bir yazım şeklinin amacı denetimi kolayca yapmaktır.

7.6. İç içe geçmiş if bildirimleri

Birden çok if bildiriminin iç içe kullanılması da mümkündür. Aşağıda bu amaçlı bir uygulama gösterilmiştir:

```
if   ‘1.mantıksal deyim'

       'bildirim grubu A'

        if   '2.mantıksal deyim'

       ‘bildirim grubu B'
       end

            1bildirim grubu C
end

            1bildirim grubu D'
```

Eğer 1. mantıksal deyim doğru ise bildirim grubu A ve bildirim grubu C uygulanır. Eğer 2. mantıksal deyim doğru ise bildirim grubu C uygulanmadan Önce bildirim grubu B uygulanır. Eğer 1. mantıksal deyim yanlış ise hemen bildirim grubu D uygulanır. Eğer 1. mantıksal deyim doğru fakat 2. mantıksal deyim yanlış ise bildirim grubu A ve bildirim grubu C uygulanır fakat bildirim grubu B uygulanmaz. Daha sonra ise bildirim grubu D uygulanır. Yukarıda görüldüğü gibi bildirim grubu D her durumda uygulanır, fakat bildirim grubu D'nin uygulanma sırası mantıksal deyimlere göre değişir. Aşağıdaki örnek incelenmelidir:

```
if  a<60
       say=say+l;
       toplam=toplam+a;
        if  b>a
           b=0;
       end
end
```

Önce a ve b değişkenlerinin birer sayı olduğu kabul edilsin. Eğer a<60 ise say değeri bir artırılır ve toplam değerine a değeri ilaVe edilir. Buna ilave olarak eğer b>a ise b değeri sıfıra eşitlenir.

Eğer a bir vektör ise a'nın 60'dan küçük olan elemanları için yukarıda açıklanan adımlar geçerli olacaktır. Eğer a, 60'dan küçük değilse döngü derhal ikinci end komutunun altındaki satıra atlar.

Eğer hem 'a' hem de 'b' vektör ise karşılaştırma a ve b'nin aynı indisli iki elemanı arasında yapılır. Eğer a yada b den bir tanesi sayı diğeri vektör ise ise karşılaştırma 'sayı* ve 'vektör' karşılaştırılmasındaki gibi yapılır.

7.7. else komutu

else komutu mantıksal deyim'in doğru (1) olması durumunda bir bildirim grubu'nun uygulanmasını, mantıksal deyim'in yanlış (0) olması durumunda ise başka birbildirim grubu'nun uygulanmasını sağlar, else komutunun hizasına mantıksal ifade yazılmaz.

```
if   'mantıksal deyim'
        'bildirim grubu A'
else
        'bildirim grubu B'
end
```

Yukarıda gösterilen bildirimde 'mantıksal deyim' doğru ise (Iojik olarak 1), 'bildirim grubu A'da yazılan işlemler icra edilir (uygulanır). Eğer mantıksal deyim yanlış ise (Iojik olarak 0), bildirim grubu B'de yazılan işlemler uygulanır. Aşağıdaki örnek incelenmelidir:

```
if a>50.
    artma=artma+l
else
    artma=artma+2
end
```

Yukarıda verilen programda eğer a sayısı 50'den büyük ise artma adlı değişken değeri 1 artırılmakta, a sayısı 50'ye eşit veya 50'den küçük ise artma adlı değişken değeri 2 artırılmaktadır.

7.8. elseif komutu

Bölüm 7.6'da tanımlanan if-else komut yapısı iç içe birden fazla kullamldığında hangi mantıksal deyim'in doğru hangisinin yanlış olduğunu anlamak zorlaşabilir. Bu durumda program mantığım daha rahat anlayabilmek için (hem programı yazan hem de programı anlamak isteyen açısından) elseif komutu kullanılır.

```
if    'mantıksal deyim 1'
        "bildirim grubu A'
        elseif   'mantıksal deyim 2'
        'bildirim grubu B'
        elseif   'mantıksal deyim 3'
        'bildirim grubu C'
end
```

Yukanda verilen yazılımda iki adet elseif komutu kullanılmıştır. İstenir ise bunların sayışım artırmak da mümkündür. Verilen elseif yapısı şöyle çalışmaktadır: Eğer mantıksal deyim 1 doğru (1) ise yalnızca bildirim grubu A icra edilir (uygulanır). Eğer mantıksal deyim 1 yanlış (0) ve mantıksal deyim 2 doğru ise yalnızca bildirim grubu B icra edilir. Eğer hem mantıksal deyim 1 hem de mantıksal deyim 2 yanlış, fakat mantıksal deyim 3 doğru ise yalnızca bildirim grubu C uygulanır. Kısaca, if döngüsü içinde mantıksal deyimin 1 (doğru) olduğu ilk satır aranır, bunu takip eden bildirim grubu uygulanır ve end'in altma inilir.

Eğer iç içe elseif yapısı içinde birden çok mantıksal deyim doğru (1) ise yalnızca ilk mantıksal deyim'i takip eden bildirim grubu uygulanır.

Eğer iç içe elseif yapısı içinde mantıksal deyim'lerden hiç biri doğru değil ise bu yapı içindeki hiçbir bildirim grubu uygulanmaz. Aşağıda else ve elseif komutlarının birlikte kullanıldığı if yapısı gösterilmiştir:

```
if     'mantıksal deyim 1'

          'bildirim grubu A'

          elseif   'mantıksal deyim 2'

          'bildirim grubu B'

          elseif   'mantıksal deyim 3 '
           ‘bildirim grubu C’
else    % buraya kadar tum mantiksal deyimler vanlis ise asagidaki bildirim grubu uygulanır
          'bildirim grubu D'
end
```

Yukanda verilen if yapısında her üç 'mantıksal deyim' de yanlış ise yalnızca bildirim grubu D uygulanır. Bu yapı içinde birden çok elseif fakat en çok bir adet else komutu kullanılabilir.

Aşağıda x değişkeninin sayı mı, vektör mü, yoksa matris mi olduğunu tespit eden bir MATLAB programı verilmiştir. Program mantığı şu şekilde oluşturulmaktadır; Önce editör ortamında boytest .m adlı bir altprogram (function) daha sonra MATLAB command window ortamında bu fonksiyonu çağıran ana program yazılmaktadır;

```
function boytest(x)
% değişkenin sayi, vektör, veya matrix olup olmadiginin kontrolü
 [m,n] = size(x);
if m==n & m==l
disp('değişken bir sayidir')
elseif m==l | n==l
disp('değişken bir vektördür')
else
disp('değişken bir matristir') end
```

Yukanda verilen altprogramı koşturan ana program aşağıda verilmiştir;

```
»a=2,  b=[3  6  7  8],  c=[l  2  3;  4  8-1;  0  7  8.1];  ('enter')
»boytest (a)
('enter')
değişken bir sayıdır
»boytest (b)
('enter')
değişken bir vektördür
»boytest (c)
('enter')
değişken bir matristir
```

Yukanda görüldüğü gibi boytest adı ile boytest. m adlı altprogram çalışmaya başlıyor sırası ile a, b ve c değişkenlerinin boyuttan test ediliyor.

Aşağıda verilen MATLAB programında ikinci dereceden bir denklemin katsayılan ekran üzerinden girildiğinde bulunan delta değerine bağlı olarak köklerin durumunu bildiren açıklamalar ekrana yazdırılmaktadır:

```
clear all     % workspace üzerindeki tüm değişkenler silinmektedir
clc                  % ekran temizlenmektedir
disp(‘Bu program 2.dereceden bir denklemin köklerini hesaplar')
disp('denklem tipi: a*x^2+b*x+c’ )
a=input(*a katsayısını giriniz');
b=input(*b katsayısını giriniz');
c=input(*c katsayısını giriniz');
delta=b^2-4*a*c;    % diskriminant hesabı yapılıyor
xl=(-b+sqrt(delta))/2*a;    % birinci kök değeri
x2=(-b-sqrt(delta))/2*a;    % ikinci kök değeri
if delta<0
    dispp(‘kokler reel değil komplekstir')
    xl
    x2
elseif delta==0
   disp(‘kökler katlıdır')
    xl
    x2                                '
else   % yukarıdaki her iki seçenek sağlanmadığında asagidaki seçenek % mutlaka sağlanır
   disp('kökler reeldir')
   xl
   x2
end
```

Aşağıda verilen MATLAB programında girilen ay numarasına bağlı olarak o ayın kaç gün içerdiği bildirihnektedir;

```
ay=input('yılın kaçıncı ayını merak ediyorsunuz,sayıyı giriniz')
 if   ay==l|ay==3|ay==5|ay==7|ay==10|ay==12
    'girilen ay 31 gun içermektedir'
    elseif   ay==2
    'girilen ay 28 gun içermektedir'
     else
    'girilen ay 30 gun içermektedir'
    end
```

Aşağıda düzgün olarak dağılmış olan rasgele değişkenli 0 ile 1 arasında 100 elemanlı bir satır vektörü üretilerek, bu vektör elemanları içinde değeri 0.6 dan büyük olan kaç tane eleman olduğunu bulan bir program yazılmıştır:

```
say=0;
A=rand(100,1);
for i=[l:100]
if A(i)>0.6
say=say+l;
      end
```

Bir banka yatırmış olduğunuz anaparaya aylık şu şekilde bir faiz uygulamaktadır: Anapara 5.000$ dan aşağı ise %2, 5.000-10.000$ (dahil) arasında ise %2.5, 10.000$ den yüksek değerler için %3 faiz uygulamaktadır. Bu çalışma koşullan altında kullanıcının anaparayı ekrandan girmesi durumunda girilen anaparaya göre bir ay sonra bankadan alacağı toplam para miktarım gösteren bir program aşağıda verilmiştir;

```
Format sfıort
anapara =input('Bankaya yatirmak istediğiniz anaparayi ($) yaziniz:  ');
 if anapara < 5000
      oran = 0.02;
      elseif anapara>=5000 & anapara<= 10000
      oran = 0.025;
else
      oran = 0.03 ;
end
toppara = anapara + oran*anapara ;
format bank
disp('Bir ay sonraki yerii para durumunuz:' )
disp( toppara )
```

7.9. for döngüsü

Bir matris veya vektör gibi birden çok sayı barındıran değişkenlerin içindeki bazı sayıları veya tümünü kullanarak birtakım işlemler yapılması gerektiğinde for döngüsü kullanılır. Bu işlemin

yapılabilmesi için komutun içinde; ele alınacak değişkenin indisi, indisin başlangıç değeri, indisin artış değeri ve indisin son değerinin de olması gerekir. Genel olarak for döngüsü;

```
for 'değişken indisi'= 'indis başlangıç değeri': 'indis artış değeri': 'indis bitiş
    değeri' 'bildirim grubu'
end
```

formunda olur. Eğer yukarıda indis artış değeri belirtümiyor ise artış değeri (default) olarak 1 alınacak demektir. Aşağıdaki Örnek incelenmelidir:

Problem 7.5

a=[0.3 2 -1 4 -5 0.1 8 -3.4 7 -2.3]

vektörünün negatif elemanlarını sayıp sonucu b adlı değişkene, sıfır ve pozitif elemanlarını sayıp c adlı değişkene atayan bir MATLAB programı yazınız.

Çözüm

```
a=[0.3    2    -1    4    -5    0.1    8                -3.4    7          -2.3];
b=0; c=0;
 for k=l:l0
if  a(k)<0
            b=b+l;
    else
            c=c+l;
    end
end
```

Yukanda verilen (editör ortamında yazılan) programda k=l'den başlayarak a vektörünün elemanları sıra ile teste tabi tutulmaktadır. Eğer negatif ise b bir artırılmakta, sıfır veya pozitif ise c bir artırılmaktadır. Bu işlem k=10 (dahil) uygulanmakta ve bitirilmektedir. MATLAB command window ortamında;

```
b=
4

c=
6
```

Elde edilir.

Eğer a vektörünün boyutu (10 elemandan oluştuğu) bilinmeseydi;

```
for k=l:length{a)
if a(k}<0
b=b+l;
```

olarak yazılabilirdi.

Eğer a (ör.3*3=9 elemanlı) bir matris olsaydı, * for k=l: 9' İfadesi uygulanırken a'nın elemanları sütun sütun taranacaktı, yani a(l,l), a(2sl),a(3,l),a(l,2),a(2)2),a(3)2),a(l,3),a(2,3),a(3,3) sırası gözetilerek test yapılacaktı.
Birden fazla for döngüsü iç içe kullanılabilir. Böyle bir durumda en içte yer alan for döngüsü daima önceliğe sahiptir. Birden fazla for döngüsü bulunan bir algoritmada, en içte yer alan for döngüsünün çevrimi bitmeden, bir üstünde yer alan for döngüsünün değişken değeri artırılmaz.

```
a=[l  2  3   4;   5   6  7   8;9  10    11    12;13   14    15   16] ;
 for k=l:4
    for m=l:4
       b(m,k)=a(k,m) ;
end
end
```

Yukandaki verilen (editör ortamında yazılan) MATLAB programmda a matrisinin devriği (transpozu) alınarak b matrisi oluşturulmaktadır. İşlem sırası şu şekilde gerçekleşmektedir;

k=l,m-I)b(I,lH(l,I)=l,m-2,b(2,ih a(l,2)=2, m-3,b(3,l)=a(l,3H,m-4,b(4,l)- a(I,4H,
k=2,m=l,b(l,2H(2,lH,m=2,b(2,2)= a(2,2)=6, m=3,b(3,2H(2,3)=7,m=4,b(4,2)= a(2,4)=8,
k=33m=l,b(l,3H(3,lH,m=2,b(2,3)- a(3,2)=10, m=3,b(3,3)=a(3,3)=ll,m=4,b(4,3)= a(3,4)=12,
k=4,m-l,b(l,4H(4,l)=133m=2,b(2J4)- a(4,2)=14, m=3,b(3,4)=a(4,3)=15,m=4,b(4,4)= a(4,4)=16,

Görüldüğü gibi k=1 için m=l,2,3,4 değerlerini almakta, yani en içteki döngünün indisi (m) son değerine ulaşmadan dıştaki döngünün indisi (k) artmamaktadır.Yukandaki dizilişe göre b matrisi;

```
b =
     1     5     9     13
     2     6     10    14
     3     7     11    15
     4     8     12    16
```

elde edilir. Yukarıda yazılan programda dikkat edilmesi gereken önemli bir nokta vardır: MATLAB'da bir matrisin başka bir matrise hatasız olarak atanabilmesi için (satır-sütun anlamında) her iki matrisinde aynı boyutta olması gerekir. Eğer satır ve sütun sayısı yukandaki örnekte aynı olmasaydı, sonuçta elde edilen b matrisi anlamsız bir matris olurdu.
f or döngüsünün özellikleri aşağıda verilmiştir:

- f or döngüsünün indisi mutlaka bir değişken olmalıdır
- Eğer üzerinde işlem yapılan matris boş matris ise döngü içinde işlem yapılmaz ve end komutunun altındaki satıra geçilir.
- Eğer matris yerine bir sayı kullanılırsa döngü bir kere uygulanır ve end komutunun altına geçer. " Eğer matris yerine bir vektör kullanılırsa vektör elemanları sıra ile işleme tabi tutulacaklardır.

H Değişken matris olduğunda tarama sütun-sütun yapılacaktır (önce birinci,sonra ikinci,...)

- f or döngüsü tamamlandığında indis en son aldığı değerde kalır.
- f or 'indis - 'başlangıç değeri': 'artış değeri': 'son değer1
  ifadesinde 'son değer-başlangıç değer* farkı artış değerine tam olarak bölünemiyor ise döngü son değere en yakın ve son değerden küçük olan artış değerini yerine getirir ve sona erer. Örnek olarak f or k=3:6:54 döngüsünde 54 sayısından önce 51 sayısı gelmektedir (3-9-15-21-27-33-39-45-51-57). İndis 57 sayışım alamayacağından dolayı (zira 57 sayısı döngünün son değerinden daha büyüktür) en son değer olan 51 sayısını alır ve döngü sona erer.

Problem 7.6

Verilen bir değişkenin en büyük elemanını ve bu elemanın bulunduğu yerin satır ve sütun numarasını bulan bir MATLAB programı yazınız.

Çözüm

Yazılacak program şu şekilde çalışacaktır; Yukarıdaki istemi yerine getiren maxbul. m adlı bir alt program (function) yazılacak ve ana program metninde maxbul komutu görüldüğünde bu alt program çalıştırılacak ve istenen amaç gerçekleştirilmiş olacaktır:

```
function [r   c   xmax] = maxbul(x)
%maksimum sayinin bulunduğu satir ve sütun
 % numarasinin elde edilmesi
[m   n]=size(x);
xmax=x(1,1);
r=l; c=l;
for k=l:m
 for g=l;n
              if x(k/g)> xmax
                   xmax = x(k,g);
                   r-k;
                   c=g;
              end
     end
end
```

Yukarıda verilen maxbul. m adlı altprogramı çağıran ana program aşağıda gösterilmiştir:

```
»A=[8];                                                    ('enter')
»B=[0.1 -8   3    7];                                      ('enter')
»C=[2     4     6;     11     12     13;     0     2.1     9.8];     ('enter')
»[r              c              ymax]              =maxbul              (A)
                                                          ('ente
r')
r=
1
```

A matrisinin içindeki en büyük elemanın bulunduğu satır sayısı 1, sütun sayısı 1 ve bu eleman değerinin ise 8 olduğu anlaşılmaktadır.

```
»[r c ymax] =maxbul (B)                    ('enter')
r=
     1
c=
     4
ymax=
        7
```

B matrisinin içindeki en büyük elemanın bulunduğu satır sayısı 1, sütun sayısı 4 ve eleman değerinin ise 7 olduğu anlaşılmaktadır.

```
>>[r c ymax] =maxbul (C)                    ('enter')
r=
     2
c=
     3
ymax=
        13
```

C matrisinin içindekten büyük elemanın bulunduğu satır sayısı 2, sütun sayısı 3 ve bu eleman değerinin ise 13 olduğu anlaşılmaktadır.

Problem **7.7**

A(x) = (x - l)(x - 3)(x - 2)(x - 8)(x + 4)(x - 5) B(x) = (x-7)(x-4)(x-8)(x-3)(x-5)

olarak verilen iki polinomun A(x)/B(x) bölümünü sıfır yapan x değerleri bulunmak isteniyor. Bu işlemi yapan MATLAB programım yazınız.

Çözüm

İki polinomun bölümünü sıfir yapan değerler aynı zamanda pay ifadesini sıfır yapan değerlere eşit olacaktır. Bu kuralın tek istisnası payı sıfir yapan kök değerleri ile paydayı sıfır yapan kök değerlerinden bazılarının aynı olmasıdır. Verilen problemde böyle bir durum ile karşılaşılmaktadır. 3, 5 ve 8 değerleri hem payı hem de paydayı sıfır yaptığı için A(x)/B(x) denkleminin kökü olamazlar. Aşağıda verilen programda pay ve paydayı sıfır yapan kök değerleri çözüm kümesinden dışarı çıkarılmaktadır. Program içinde kullanılan C adlı vektörün elemanları istenilen çözüm kümesine ilişkin kökleri vermektedir.

```
k4=0;
A=[l328 - 4 5 ] ;
B=[74835 ] ;
```

```
forkl=l:length{A);
k3=0;
   fork2=l:length(B);
   ifA(kl} ~= B(k2);
         k3=k3+l;
       else
   end
   end
       ifk3==length(B>;
        k4=k4+l;
        C(k4)=A(k)
         else end
```

Yukarıdaki program uygulandığında aşağıdaki değerler elde edilir:

```
» C
=
1       2       -4
```

Problem 7.8

```
B=[-1.3317-2i0.6+0.7i   2 10 0.6-0.7i]
```

satır vektör matrisinde 10 adet eleman bulunmaktadır. Bu elemanların x^3 - $6x^2$ +1 İs - 6 = 0 denkleminin
köklerinden bir yada birkaçım içerip içermediği kontrol edilecektir. Verilen denklem 'denklem.m' adlı
bir f unction dosyasında saklanacaktır. Ana program B elemanlarını tek tek okuyacak, okuduğu değerin alt
programdaki denklemi sağlayıp sağlamadığını kontrol edecek, eğer bu değer denklemi sağlıyor ise ana
program içinde değerindeki eleman verilen denklemin bir köküdür' ifadesi
yazılacak ve bu işlem tüm B elemanları için bir döngü içinde tekrarlanacaktır.

Çözüm

```
format compact     % çıkıştasatırları birbirlerine yakınlaştırır
B=[-1.3064 3 1 7 -2 i 0.6532+0.7281 i  2   10  0.6532-0.7281i ] ■ ;
fork=l:10

      a=B(k,l);
      b=denklem (a);
      if b==l;
      disp (a)
disp('değerindeki eleman verilen denklemin bir kokudur')
end
```

Aşağıdaki MATLAB dosyası denklem.m adı ile kaydedilecektir, denklem.m adlı MATLAB dosyası bir 'alt program' dosyasıdır.

```
function hedef=denklem(y) ;
C=[l -6 11 -6];
d=roots(C);
 for t=l:3
     if abs(y-d(t,1)}<=10*eps;        % açıklamaya bakınız
     hedef=1;
          break     % programı durdurur
     else
          hedef=0;
     end
     end
```

Açıklama:

Yukarıdaki program satırlarında,

İf abs(y-d(t,l))<=10*eps;

satın yerine if y= =d(t,l)

satırı kullanılmalıydı. Fakat MATLAB ortammda iki sayının birbirine eşit olabilmesi oldukça zordur, zira MATLAB'ın sayı duyarlılığı oldukça yüksektir. Bu nedenle kullanıcıya eşit gibi görülen iki sayı duyarlılık dolayısı ile MATLAB arka planında eşit olmadığı için istenilen hedefe ulaşılamaz. Yukarıda bu problemi halletmek için sıfıra yakın bir sayı olan 10*eps değeri kullanılmıştır, (eps; MATLAB ortamında tanımlı ve değeri 2.2204e-016 olan çok küçük bir sayıdır)

Problem 7.9

JPEG formatında 2 adet resim MATLAB ortamında iki adet matrise dönüştürülmüştür. Bu iki adet resmin birbirlerinin aynı olup olmadığı bir MATLAB programı yardımıyla test edilecektir. Bu iki resmin boyutları aynı değilse 'Bu iki resim farklıdır' yazdırılacaktır. Eğer bu iki matrisin hem boyutlan hem de tüm elemanlan aynı ise 'Bu iki resim aynıdır' yazdırılacaktır. Eğer matris boyutlan aynı ve iki matrisin en fazla 1 adet elemanı farklı ise 'Bu iki resim aynı fakat bozulma var' yazdırılacaktır. Eğer matris boyutları aynı fakat iki matristeki farklı eleman sayısı 1 den fazla ise 'Bu iki resim birbirlerinden farklıdır' yazdınîacaktır.

Çözüm

Birinci resmin matris hale dönüştürülmesi sonunda elde edilen matris Al, diğeri ise A2 olsun.

```
Al=[0 1 3;4 5 6];
 A2=[l 2 3;4 5 6];
```

```
 [satiri   sutunl]=size(Al);
[satir2   sutun2]=size(A2);
if ((satirl~=satir2)|| <sutunl~=sutun2))
'Bu iki matris farklidir'
else
end
%

if ( (sâtirl==satir2) &(sutaml==sutun2) &(Al==A2) )
'Bu iki matris aynidir'
else
end
% ------_ ----------------------------------
if ((satirl==satir2)&&(sutunl==sutun2))
say=0;
      for   kk=l:satir1
      for
      mm=l:sutunl
            if Al(kk,mm)==A2(kk,mm)
                  say=say+l;
                  else
            end
            end
            end

else
       'Bu iki matris farklidir'
end
% ----------------------------------------
sayl=satirl*sutunl;
      if (abs(say-sayl)==1)
            'Bu iki resim ayni fakat bozulma var'
      else
      end
% ----------------------------------------
      if (abs(say-sayl)>1)
            'Bu iki resim farklıdır'
     else
end
```

Problem 7.10

a adlı bir satır vektörünün tüm elemanlarım tarayarak en küçükten en büyüğe doğru b adlı vektöre atayan bir MATLAB programı yazımz. b vektörünün elemanları ilk eleman a vektörünün en küçük elemam olacak şekilde sıralanacaktır.

Çözüm

```
a=[l 2 3 4 5] ;
for   m=l:size(a,2)      % a vektörünün boyutu
      [y k]=min(a) ;
     a(k) =inf;
      b(m)=y
end
```

Problem 7.11

Yukanda verilen V değerini bulunuz.

Çözüm

MATLAB Editor ortamında yazılan;

```
top=0;
for m=l:10^300:2 % inf olarak 10 üzeri 300 sayisini aldik
a=(400/pi)*(l/m)*(sinh(m*pi/4)/(sinh(m*pi)));
top=top+a;
end
top
```

dosyanın çalıştınlması sonunda elde edilen sonuç Command Window ortamında;

```
»top =

          9.5770
```

olarak elde edilir.

7.10. break komutu

For döngüsü içinde değişken son değerine ulaşmadan döngüden çıkmak için break komutu kullanılabilir.

Aşağıda verilen dosyada, A matrisinin içindeki ilk 0 aranmakta, sayı yakalandığında bu sayının bulunduğu satır ve sütun numarası saptanıp iç içe olan for döngüleri durdurulmakta ve hesaplama 2. end komutunun altından devam etmektedir. Programın en altında ise 0 sayısının bulunduğu satır ve sütun numaralan yazdırılmaktadır.

```
A=[l 5 6;7 -3 9;-4 0 8];
      for t=l:size(A,l) %t; l'den A'nın satır sayısına kadar artıyor
      for g=l:size(A,2) %g; l'den A'nın sütun sayısına kadar artiyor
      if A(t,g)==0
```

```
        satir„numarasi=t;
        sutun_numaras i=g;
        break;
        end
        end
end
    satirj_numara
    si=t;
    sutun_numara
    si=g;
```

Not: Yukanda verilen programın en başına tic, en sonuna ise toe yazılır ve program tekrar çalıştınlırsa elapsed_time=0.16 elde edilir. Bunun anlamı; tüm işlemlerin 0.16 saniye içinde bitirildiğidir.

Eğer yukarıda verilen programda break komutu kaldırılır, program tekrar çalıştırılır ve zamana tekrar bakılırsa, 0.22 saniye elde edilir. Böylece break komutu ile program kesilerek kullanıcıya zaman tasarrufu sağlanmaktadır.

7.11. continue komutu

Yukarıda bahsedildiği gibi bir döngü sona erdirilmek istendiğinde break komutu kullanılmaktadır. Bazen döngünün durdurulması yerine yalnızca döngünün o an geçerli olan değişkeni bir atlatıp döngüye devam etmek istenebilir.

Aşağıda verilen programda B matrisinin tüm elemanları 2'ye bölünmekte fakat mutlak değeri 6'dan büyük olan A matrisi elemanlarına bu işlem uygulanmamakta, bu durum ile karşılaşıldığında continue komutu ile döngüde o an geçerli olan değişkenin aldığı değer atlanmakta ve döngüye devam edilmektedir. Son olarak A matrisinin son hali ekrana y azdırılmaktadır..

```
B=[2 5 11;7 -3 9;-4 0 8] ;
 for t=l:size{B,l)
 for g=l:size(B,2)
       if abs{B(t,g))>6
       continue
       end
   B(t,g)=B(t,g)/2;
   end
end
    B
```

Yukandaki programın uygulanması sonunda elde edilen B matrisi değeri aşağıda verilmiştir:

```
» B
=
1     2.5     11
```

```
7 -1.5    9
-2        0        8
»
```

Aşağıda verilen MATLAB programında A vektörünün sıfıra eşit veya sıfırdan küçük değerli elemanları için (10 tabanına göre) logaritma işlemi yapılmamakta (bu şarta uygun sayılar A vektörü içinde sayılar aynen bırakılmakta), yalnızca A'nın pozitif değerli elemanları için A vektör elemanlarının logaritması alınmaktadır:

```
A=[0    1    -5    2    8    -7];
for n=l:length{A)
      if A(n)<=0
      continue
      end
A(n)=logl0(A(n))
end
A       % A vektörünün pozitif elemanlarının
        % logaritmaları alındığında elde edilen yeni A vektörü
```

Yukarıda verilen programın çalıştırılması sonunda elde edilen A vektör eleman değerleri aşağıda gösterilmiştir:

```
» A  =
0   0   -5.0000   0.3010   0.9031   -7.0000
```

7.12. return komutu

Ana programdan alt programa dallanıp alt program içinde istenilen şart gerçekleştiğinde alt programdaki döngüden çıkıp tekrar ana programda kalındığı yerden devam edilmek isteniyor ise alt program içinde return komutu kullanılmalıdır.
Aşağıda bul .m 'adlı alt program verilmiştir;

```
function [satir_.no, sutun_.no] =bul (B)
for  t=l:Sİze(B,l)
  for g=l:size{B,2}
    if B(tfg)==0
      satir_no=t;
      sutun__no=g;
      return;
    end
  end
end
```

Yukanda verilen Command Window ortamındaki iki program satırının çalıştırılması ile aşağıdaki sonuçlar elde edilir:

```
»
```

```
Satir_  no =
              3
Sutun_no =
              2
```

7.13. error komutu

Kullanıcı bazen programı belli koşullan gözeterek yazar. Eğer program içinde kullandığı değerler bu koşullan sağlamaz ise program çalışabilir fakat sonuçları doğru olmayabilir. Kullanıcı, ileride bu programı çalıştırdığında bu koşulan unutabileceğim düşünerek yazdığı program içine bir komut yerleştirerek hem programın durmasını hem de neden durması gerektiğine dair mesajın (hatırlatmanın) ekrana yazılmasını istediğinde (veya buna benzer amaçlar için) error komutunu kullanabilir.

Örnek olarak kullanıcı, A adlı bir vektör içine pozitif doğru akım bilgileri girmekte ve yazdığı ortalama_bul.m adlı alt program ile A vektörünün ortalamasını hesaplamaktadır. Bilindiği gibi pozitif doğru akım değeri asla pozitif değerden negatif değere geçmez. Eğer A vektörü içinde herhangi bir eleman değeri negatif ise kullanıcı error komutu yardımı ile hem programı durdurabilir hem de durdurma sebebini ekrana yazdırabilir. Yine bilinmelidir ki, içinde negatif sayı barındıran bir vektörün ortalaması (mean komutu yardımı ile) hesaplanabiimektedir. Diğer bir ifade ile böyle bir programda error komutu kullanılmadığında herhangi bir hata ile karşılaşılmaz. Bu programı kesmek kullanıcının bir tercihidir.

```
» A=[1 5 6 7 -3 9 4 0 8] ;                    (‘enter')
» ortalama=ortalama_bul (A)                   ('enter')
```

Aşağıda ortalama_bul. m adlı alt program verilmiştir;

```
% Bu program içinde negatif değer barindirmayan bir işaretin ortalamasini bulmak için kullanilir

function d=ortalamâ_bul(B)
for t=l:length(B);
   if B(t)<0
    error {'matris elemanlarından hic biri negatif olamaz')
    end
end
d=mean(B);
```

Yukarıda verilen iki program satırının uygulanması sonunda elde edilen sonuçlar aşağıda verilmiştir. Görüldüğü gibi programın çalışması error komutu kullanılarak durdurulmakta ve ortalama hesabı yaptırılmamaktadır.

```
»
??? Error using ==> ortalama_bul
matris elemanlarından hic biri negatif olamaz
```

7.14. warning komutu

error komutunda kullanıcının istediği mesaj ekrana yazılmakta ve program durdurulmaktadır, warning komutunda ise kullanıcının istediği mesaj ekrana (uyarı olarak) yazılmakta fakat programın çalışması devam ettirilmektedir. Bu komutun uygulamasına örnek olarak yukarıda verilen ortalama_bull.m adlı alt programda error komutu yerine warning yazılıp uyan metni değiştirilebilir. Aşağıdaki alt program incelendiğinde A vektörünün ortalaması alınmakta fakat uyarı ifadesi ile de kullanıcı uyarılmaktadır:

```
function d=ortalama__bull (B)
for t=l:size(B,l)
  for g=l:size(B,2)
    if B(t,g)<0
          warning {['matris elemanlarindan hic biri negatif olamaz'... ' fakat yine de ortalama
               alinmaktadir' ] )
        end
   end
end
d=mean (B) ;
```

Aşağıda verilen satırlar uygulandığında;

```
» A=[l 5 6 7 -3 9 4 0 8] ;                    (‘enter')
» ortalama=ortalama_bull (A)                  ('enter')

Warning:   matris  elemanlarından hic biri negatif  olamaz   fakat  yine de ortalama alınmaktadır
ortalama =
                  4.11111111111111
```

elde edilir.

7.15. eval komutu

Bu komut MATLAB ortamında çeşitli biçimlerde kullanılabilir. Aşağıda verilen örnekler incelenmelidir:

```
»x=0:l:3;                                     ('enter')
»y=eval(‘2*x.^2+sin(2*x)')                    ('enter')
y =
        0   2.9093 7.2432   17.7206
```

Yukarıdaki satırlarda x değerleri iki tırnak içinde verilen fonksiyonda yerine konularak elde edilen değerler y vektörüne atanmaktadır. Yukarıdaki işlem;

```
»y=eval ('2*x. ^2+sin (2*x) ',0:1:3);   ('enter')
```

komut satırı ile de gerçekleştirilebilir. Aşağıda eval komutunun bir başka uygulaması

```
for n=l:3
```

```
eval (['A' num2str(2*n) '= [n ;n-l ;2*n]']) % köşeli paranteze dikkat!
end
```

Yukarıda verilen program uygulandığmda aşağıdaki sonuçlar elde edilir:

```
>>
A2 -
        1

        0

        2
A4 =
        2

        1

        4
A6 =
        3

        2

        6
```

Açıklama: num2str (x) komutu ile x sayısı MATLAB ortamında karakter gibi değerlendirilir. Bu nedenle yukarıda verilen program satırında (2*n) sayısı A adlı karakterle birleştirilebilmektedir. eval komutunun bulunduğu satırda yer alan [n ;n-l ;2*n] vektör matrisi n döngüsü ile değer değiştirmektedir.

```
karakter='<=';
A=[-l 4 6;11 -8 -6];
for n=l:2
  for m=l:3
     if eval ([num2str ((A(n,m))) karakter num2str(0)]) % köşeli paranteze dikkat!
     A(n,m)=abs{A{n,m)) ;
     end
   end
end
A
```

Yukarıda verilen programda ise A matrisinin elemanları içinde O'a eşit veya küçük olanın mutlak değeri alınmakta ve son olarak yeni hali ile A matrisi ekrana yazdınlmaktadır. Programın uygulanması sonunda elde edilen A değeri aşağıda gösterilmiştir;

```
A =
 146
     11     8     6
```

>>
Yukanda verilen programdaki eval komutunun bulunduğu satır;
i f eval ([‘ A(n,m)' ' <=' '0']) % köşeli paranteze dikkat!
biçiminde de yazılabilirdi. Bu durumda da A matrisi aynı değeri alırdı.

7.16. feval komutu \

feval ('fonksiyon adı', değerler) : 'değerler' bölümünde yer alan sayının aldığı değeri 'fonksiyon adı' kısmında yazılı olan fonksiyona koyarak hesaplatan bir MATLAB komutudur.

' fonksiyon adı' kısmında yer alan fonksiyon MATLAB arka planında tanımlı (sin,cos,tan, gibi) fonksiyon ise;

```
»b= feval(@sin,pi/3)                    ('enter')
b =
      0.8660
```

program satırlarında görüldüğü gibi pi/3 açısı (radyan olarak) sin(x) ifadesinde x yerine konularak sin(pi/3) değeri hesaplanmakta ve sonuç b değerine atanmaktadır. Burada görüldüğü gibi iki adet tırnak işareti yerine @ işareti de kullanılabilir, feval komutunun alt program uygulamalan ile ilgili örnekler ise Bölüm 10'da verilmiştir.

7.17. Döngü süresini kısaltmak

Genel olarak MATLAB programlarında 'döngülerden kaçınmak gerekir. Döngüler programın icra süresini önemli sayılacak miktarda artırırlar. MATLAB'm alt yapısı, kullanılan vektör veya matrislerin boyutlarının büyütülmesini mümkün kılan, diğer bir ifade ile boyutların büyütülmesinden dolayı icra süresinin fazla artış göstermediği bir özelliğe sahiptir.

Örnek olarak sinüs dalgasının genliğinin üretildiği aşağıdaki MATLAB programı icra edilsin:

```
% 5000 uzunluğa sahip sinuzoidal
for t=l:5000
y(t) = sin(2*pi*t/10);
end
```

Yukarıda verilen programın icra süresi (kişisel bilgisayar için) t=7 . 91 saniyedir. Aşağıdaki programın;

```
% 10000 uzunluğa sahip sinuzoidal
for t=l:10000
y(t) = sin(2*pi*t/10);
end
```

icra süresi ise (kişisel bilgisayar için) t=28.56 saniyedir. Görüldüğü gibi uzunluk iki katına çıkmasına rağmen icra süresi yaklaşık 4 katına çıkmaktadır (geometrik artış).

Programın icra süresini azaltmak (hızı artırmak) için döngüden önce bir dizi oluşturmak yeterlidir. Böylece her bir döngü adımı için y uzunluğunu tekrar artırmak gerekmeyecektir. Aşağıdaki program için;

```
% 10000 uzunluğa sahip sinuzoidalin hesabinda
% vektör kullanimi ile zaman kazanimi
y = zeros(1,10000);
for t=l:l0000
y = sin(2*pi*t/10);
end
```

icra süresi ise (kişisel bilgisayar için) t=2.03 saniyedir. Yukarıda (zeros komutu kullanılarak) 10000 adet elemanı sıfır olan bir y vektörü oluşturularak döngünün her seferinde başa dönüp yer açmasından kaynaklanan süre artırıcı etki azaltılmıştır. Yukanda verilen programın sağladığı süreden daha da iyi bir süre tasarrufu sağlayan diğer bir program ise aşağıda verilmiştir:

```
% sinuzoidal olarak değişen sayı üretmenin daha iyi bir yolu
%
t = 1:10000;
y = sin(2*pi*t/10);
```

Yukanda verilen programın icra süresi İse (kişisel bilgisayar için) t=0.06 saniyedir.

7,18. while döngüsü

while döngüsü, 'mantıksal deyim' doğru (1) olduğu sürece bildirim grubu'nu icra etmeye (uygulamaya) devam ettiren, mantıksal deyim yanlış (0) olması durumunda ise döngüyü sona erdiren bir MATLAB komutudur. Eğer mşntıksal deyim daima doğru (1) olursa while döngüsü sonsuz çevrime girecek ve döngü sona ermeyecektir. Böyle bir durumla karşılaşıldığında (Ctrl + C) ^c tuşuna basılarak döngü durdurulmalıdır. Aşağıda while döngüsünün genel biçimi gösterilmiştir:

```
while   'mantıksal deyim'
          'bildirim grubu'
end
```

Not:

```
while a<k
…..

end
```

şeklindeki bir yazılımda döngü a>k oluncaya kadar

devam eder.

while a>k
…..
end

şeklindeki bir yazılımda döngü a<k oluncaya kadar devam eder.

Aşağıda verilen programda b>a koşulu (mantıksal deyim) daha başlangıçta sağlanmadığı için döngü içine girmeden döngü sona ermektedir.

```
a=4
b
=
l
while b>a
b=b+1;
end
```

Yukandaki program satırları uygulandığında aşağıdaki değerler elde edilir:

```
>>
a =
        4
b =
        1
```

Aşağıdaki program satırlan incelenmelidir:

```
a=4;

b=1;
while b<a
        b=b+l
end
```

Yukarıdaki program satırları uygulandığında aşağıdaki değerler elde edilir:

```
>>
b=
2
b=
   3 b=
   4 % b=4 için uygulanmadi. b=3 için uygulandiginda 1 ilave edildi 4 oldu
```

Aşağıda verilen programda verilen bir a vektörünün içindeki ilk negatif elemanı tespit eden ve bu elemanın vektörün kaçıncı elemanı olduğunu (editör ortamında) yazan bir MATLAB programı verilmiştir:

```
a=[l.l  5.6    4.9    6.8    5.4    -2.3    9.2    -8.2];
k=l;
while a(k)>0
k=k+l;
end
disp('a nin ilk negatif elemani')
a(k)
disp('ilk negatif eleman a nin')
k
disp('inci elemani')
»
a nin ilk negatif elemani
ans =
-2.3000 ilk negatif eleman a nin
k =
6
inci elemani
»
```

elde edilir. Aşağıda verilen MATLAB yazılımında,

$$toplam = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$$

olarak verilen bir
toplama dizisinin sonucu 1.6 oluncaya kadar devam ettirilmek isteniyor. Bu toplam sağlandığında n sayısının aldığı değer ve toplam değeri bulunuyor:

```
n=l;
toplam=0;
while   toplam<l. 6
toplam=toplam+n^(-2);
n=n+l; end
toplam
n-1   % n-1 yazılmasının nedeni toplam satırının altında
     % toplama işleminden sonra n sayısının 1 adet
       % artırılmasıdır.
```

Yukarıdaki program çalıştırıldığında aşağıda verilen sonuçlar elde edilir.

```
>>
toplam =
1.6005
ans=
22
```

7.19. while-break döngüsü

while-end döngüsünde 'mantıksal deyim* doğru (1) olduğu sürece bildirim grubu'nu icra etmeye (uygulamaya) devam ediliyordu ve bu koşul sağlandığı sürece döngünün dışına çıkmak mümkün değildi, while-break döngüsü ise döngüden istenildiğinde çıkış imkanı

vermektedir. Bu komutta, döngü içine yerleştirilen break komut satırına gelindiğinde döngü sona erer.

Problem 7.12
İkinci dereceden (ax2+ bx + c) bir polinomun katsayıları ve x değeri girildiğinde polinomun değerini hesaplayan bir program yazınız. Eğer a, b, c ve x değeri aynı anda sıfır olursa program sona erecek, aksi halde yeni katsayılar ve x değerleri girildiği sürece program çalışmaya devam edecektir. Eğer x değeri karmaşık (kompleks) bir sayı ise yine program sona erecektir.

Çözüm

```
%   ax^2 +bx + c  polinom değerinin hesaplanmasi
'polinom   ax^2+bx+c   formundadir'
a=l; b~l;  c=l;
x=0;
while a~=0 | b~=O | c~=0 | x~=0
'eger a=b=c=x=0 olduğundan dolayi program sona erdi'
a = input('a katsayisini gir:   ');
b = input('b katsayisini gir:  ');
c = input{'c katsayisini gir:   ');
x = input('x değerini gir:  ');
if imag(x)~=0,
'x değeri karmasik sayi olduğundan dolayi iterasyon sona erdi'
break
end
polinom = a*x^2 + b*x + c; 'polinomun değeri'
   disp (polinom)
end
```

Yukarıda verilenprogramda;

```
if  imag(x)~=0,
```

satırının doğru (1) olması durumunda break komutu ile döngü dışına çıkılmaktadır. Aksi halde, döngü a,b,c ve x değerinden bir tanesi sıfıra eşit olmadığı sürece yeni katsayılar ve x değerlerini istemeye devam edecektir.
Aşağıda verilen MATLAB programında kullanıcının belirleyeceği kadar sayının tersi alınmaktadır. Kullanıcının belirlediği sayı kadar ters alma işlemi gerçekleştirildiğinde kullanıcıya tekrar işleme devam etmek, isteyip istemediği sorulmaktadır. Devam isteği *e' harfi ile hayır isteği ise 'h' harfi ile belirtilmektedir. Bu iki harf sayısal (nümerik) olmadığı, diğer bir ifade ile harf (string) olduğu için, input komutu içinde bu durum 's' harfi ile bildirilmektedir, 's' harfi kullanılmadığı takdirde hata ortaya çıkacaktır, 'e' seçeneği kullanıldığı sürece program kullanıcıdan sayı istemeye devam etmekte, lh' seçeneği kullanıldığında ise program sona ermektedir. Eğer kullanıcının girdiği sayı sıfır ise kullanıcıya hata mesaj ı verilerek program durdurulmaktadır.

```
x=input('Kaç adet sayının tersini almak istiyorsunuz?');
```

```
for m=l:x
```

```
y=input('tersini almak istediğiniz sayıyı giriniz= ');
if y==0
    error ('sıfır sayısının tersi tanımsızdır')
    end
disp('girdiğiniz sayının tersi');
l/Y
if m==x
    disp('işleme devam etmek istiyor musunuz');
    devam=input (' evet ise "e", hayır ise "h" harfine basınız ',"s');
    while devam=='e'
        x=input('sayıyı giriniz= ');
        if x==0
      error {'sıfır sayısının tersi tanımsızdır')
      end
        disp('girdiğiniz sayının tersi');
        l/x
        disp{'hala devam etmek istiyor musunuz?');
       devam=input (' evet ise "e", hayır ise "h" harfine basınız ',"s');
        if devam=='h'
                input('hesaplama sona ermiştir')
                break
               else
            end

    end
end
end
```

7.20. switch- şartlı deyimi

Bu komut ile program içinde dallandırma yapılır. Kullanıcının, belli durumlar için sadece belli ifadelerin bulunduğu ifadeler bloğunun uygulanmasını istediği durumlarda, bu komut tercih edilebilir. Bu durumlar dışarıdan girilen değişkenin değişik özelliklerine göre belirlenir:

```
switch   (durum)
case     (durum 1),

ifade 1
ifade 2
….                    1. ifadeler bloğu
….
….
ifade n case

(durum 2),
```

```
ifade 1
ifade 2
…                    2. ifadeler bloğu
…
…
ifade n
otherwise,
ifade 1
ifade 2
…                    3. ifadeler bloğu
….
….
ifade n


end
```

Aşağıdaki örnekte, sayı olarak 1-10 arasında bir rakam girildiğinde girilen rakamın tek mi yoksa çift mi olduğunu belirten uyan yazısı ile karşılaşılmaktadır:

```
sayi=input('1-10 arasında bir sayi giriniz=');
switch {sayi}
case {1,3,5/7,9},   'sayi tek'
case {2,4,6,8},  'sayi cift'
otherwise,  'sayi 1-10 aralıginin disinda'
end
```

Aşağıda verilen MATLAB programında öğrencinin aldığı notlara göre basan notu harf ile gösterilmektedir. Eğer öğrenci 50'nin altmda bir puan almış ise başansız olmaktadır

```
aldiginot==input('1 ile 100 puan arasında bir not giriniz');
switch(aldiginot)
case{50,51,52,53,54,55,56,57,58,59,60}
'E'
case{61,62,63,64,65,66,67,68,69,70}
'D'
```

```
case {71,72,73,74,75,76,77,78,79,80}                                    'C'
case {81,82,83,86,85,86,87,88,89,90} 'B' case{91,92,93,96,95,96,97,98,99,100}
'A'
    otherwise
disp('basarisiz')
end
```

Aşağıda verilen MATLAB programında girilen açı değerine göre (açı; 90,180,270 ve 360 derece olmamak koşulu ile) bulunduğu (trigonometrik) bölge numarası ekrana yazdınlmaktadır:

```
aci= input('aci değerini giriniz')
switch   fix{aci/90)     %acinin bölge numarasi hesaplanmaktadir
case 0
disp(*aci 1. bölgededir')
case 1
disp(vaci 2. bölgededir')
case 2
disppaci 3. bölgededir')
case 3
dispt'aci 4. bölgededir')
end
```

Problem 7.13

$$u(t)=\begin{cases}0 & t<0\\ 1 & t\geq 0\end{cases}$$

$$u(t)=\begin{cases}0 & t<0\\ 1 & t\geq 0\end{cases}$$

t= -20:20 aralığında basamak fonksiyonunu MATLAB ortamında oluşturunuz,

Çözüm
»t=-20:0.1:20; »u=zeros (size(t)); >>u(find(t>=0))=1;

Yukarıda zeros komutu ile tüm t değerleri için u=0 yapılmaktadır. Daha soma t > 0 için u=l yapılmaktadır, find komutu ise daha önce tanıtılmıştı.

Problem 7.14

Gürültü işaretini ortalama-örnekleme yaklaşımı ile modelleyerek filtre ediniz.

Çözüm

Daha önce (bkz.Bölüm 6-gürüItü işaretinin simülasyonu örneği) gürültü işaretinin 'rasgele sayı üretimi' yaklaşımı ile nasıl modellendiği gösterilmişti. Burada ise giriş x(t)-rasgele sayı üretme yöntemi ile elde edilen- fonksiyonundan 3 adet değer alıp bunun ortalamasını çıkış fonksiyonu y(t) ye taşıyan bir modelleme yapılması istenmektedir. Böylece daha çok işaret girişinden daha az sayıda çıkış üretilerek bir nevi filtre işlemi yapılmaktadır. Çıkış fonksiyonunun giriş fonksiyonu emsinden ifadesi (ortalama-örnekleme);

$$y(k) = \frac{1}{3}[x(k) + x(k-l) + x(k-2)]$$

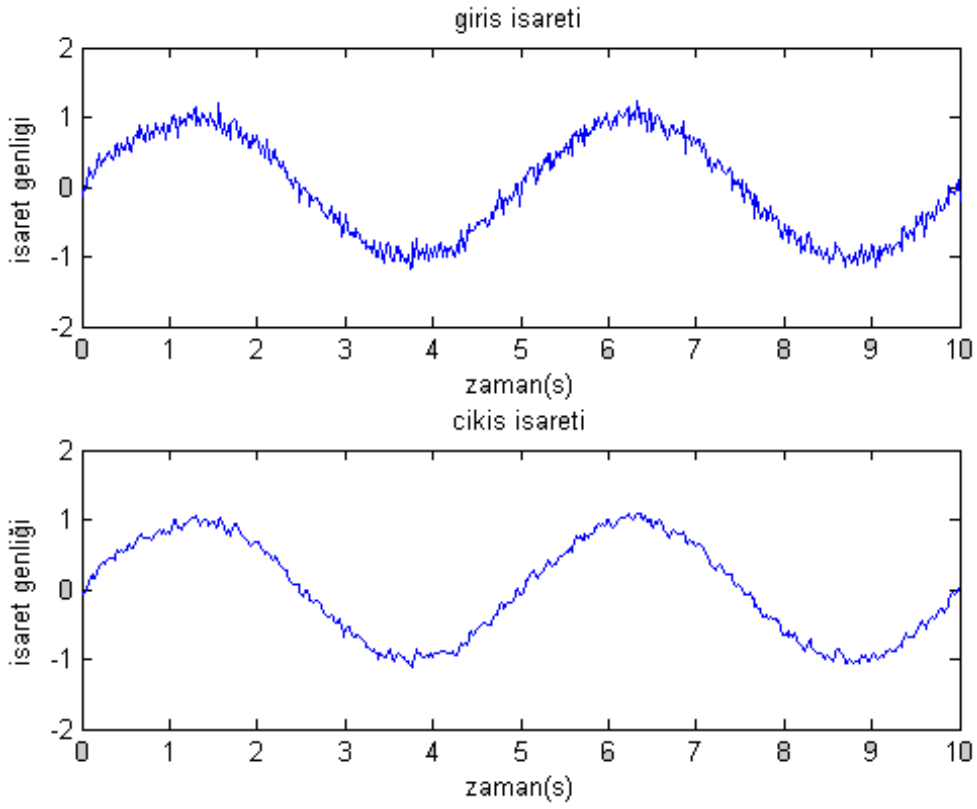
olmaktadır. Yukarıda verilen ifadede *k' örnekleme adımıdır.

```
% Gurultu işaret üretimi ve filtre edilmesi
t = linspace(0, 10,512);          % zaman ekseni
s = sin(2*pi/5*t);        % işaret
n = 0.1*randn(size(t)) ;          % gurultu, std dev 0.1
x = s + n;        % işaret + ses
disp('giris işaret gurultu oranı (IGO), dB')
% std-standart sapma
iGOin = 20*logl0(std(s)/std(n)) % giriş (IGO)-işaret giriş oranı, dB y = zeros(size(t));    % cikis işaretinin ilk değerlerinin üretimi
% filtreleme işlemi
%
y(1)  = x(l);
y(2)  = (x(2)+x(l))/2;
for k = 3:length(t);
          y(k) = (x(k)+x{k-l)+x(k-2))/3;
end

%
% filtre edilmiş işaretin analizi
%

disp('cikis isaret-gurultu orani (IGO), dB')
```

```
iGOout = 20*logl0{std(s)/std(y-s))   % cikis IGO, dB
%
% işaret çizimi
%
subplot(2,1,1) ,plot(t,x)
xlabel('zaman (s)'),ylabel('işaret genliği '),title('giriş işareti')
subplot(2,1,2),plot(t,y)
xlabel('zaman {s}'),ylabel('işaret genliği'),title('cikis işareti')
```

Yukarıda verilen MATLAB programının çıkış eğrilerine ilişkin ekran görüntüsü şekil 7.5'de verilmiştir. Programın uygulanması sonunda MATLAB Command Window ortamında elde edilen değerler ise aşağıdadır:

```
»
giriş işaret gurultu orani (IGO), dB
IGOin =
17.4577
cikis isaret-gurultu orani (IGO), dB
IGOout =
21.7848
```



Problem 7.15

a, b ve c adlarında aynı boyutta 3 adet satır vektörünün aynı adresteki elemanları birbirleri karşılaştırılacaktır. Karşılaştırma boyunca A, B ve C adlarında (a, b ve c ile aynı boyutta) üç vektör ortaya çıkacaktır, a, b ve c vektörlerinin aynı adreslerdeki elemanları birbirleri ile karşılaştırıldığında büyük olan eleman A'ya küçük olan eleman B'ye, üçünün ortalaması ise C'ye yazılacaktır. Yukarıdaki işlemi yapan MATLAB programım yazınız.

Çozum

```
a=[ 0   -2   4     6];
b=[-l    3   11   -1];
c=[4     0  -2   03]; M=[a;b;c],
A=max(M)
B=min (M)
C=mean{M)
```

Yukanda verilen MATLAB programının çalıştırılması sonunda elde edilen matris değerleri aşağıda verilmiştir:

```
 M =

0       2       4       6
-1      3      11       1
4       0       2       0

       A =
        4       3      11     6
       B =
        -1      -2      -2        -1
       C =
        1.0000     0.3333     4.3333     1.6667
```

Problem 7.16

Öyle bir MATLAB programı yazın ki, hangi dereceden olursa olsun bir polinomun tüm köklerini hesaplayabilsin. Yazılan program içinde polinomun katsayıları vektör olarak verilmeyecek, ekran üzerinden klavye ile girilecektir. Yazılan programda aşağıdaki satırlar da bulunacaktır. Program kullanıcının belirlediği kadar denklemi ard arda çözebilecektir. Aşağıda yazılacak programa ilişkin ara satırlar gösterilmiştir:

```
disp('kac adet denklem koku bulmak istiyorsunuz?'}
disp('polinomun en buyuk derecesini gir')
disp('en buyuk dereceden başlayarak sıra ile katsayilari giriniz')
disp('inci denklemin kökleri')
```

(Kullanıcı, dilediği kadar polüıomun köklerini ard arda bulmak istemektedir. MATLAB programında kullanıcının karşısına önce "kaç adet denklem bulmak istiyorsunuz?'1 sorusu çıkacaktır. Daha sonra sıra ile 'en büyük polinom' derecesinden başlayarak polinomun katsayılan sıra ile klavye kullanılarak girilecektir. Ekranda kaçıncı denklemin kökleri olduğunu belirten açıklama da yer alacaktır. Bu işlem tüm denklemler bitene kadar devam edecektir)

Çözüm

```
»
```

```
Disp(' kaç adet denklem koku bulmak istiyorsunuz?')
aa=input('  ');
n=0;
while n<aa n=n+l;
disp('polinomun en büyük derecesini gir') A=input('') ;
disp('en büyük dereceden başlayarak sıra ile katsayıları giriniz')
c=size(A,1);
for k=l:A+l
c(k)=input(' ')
end
disp('inci denklemin kökleri ')
roots(c)
end
```

BÖLÜM 8

VEKTÖR VE MATRİS İŞLEMLERİ

Matris iki boyutlu bir diziliştir.

A=[3.5] Matris işlemlerinde bir sayı
olarak,

vektör (bir satır-iki sütun);

A_1=[-3.4 9.6] ; $A_1(1,l) = -3.4$; $A_1(1,2) = 9.6$

veya (iki satır-bir sütun);

$A_2 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

larak, **matris** ise (iki satır-iki sütun);

$$A_3 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

8.1. Vektörler

Genellikle vektörler sütun vektörü olarak gösterilir.

$$\mathbf{X}=\begin{bmatrix} x1 \\ x2 \\ x3 \\ . \\ . \\ . \\ xn \end{bmatrix}$$

8.1.1. İki vektörün toplamı ve **farkı**

$$X=\begin{bmatrix} x1 \\ x2 \\ . \\ xn \end{bmatrix} \quad Y=\begin{bmatrix} y1 \\ y2 \\ . \\ yn \end{bmatrix} \quad X+Y=\begin{bmatrix} x1+y1 \\ x2+y2 \\ ..... \\ xn+yn \end{bmatrix} \quad X-Y=\begin{bmatrix} x1-y1 \\ x2-y2 \\ ..... \\ xn-yn \end{bmatrix}$$

8.1.2. İç veya nokta çarpım

x ve y adlı iki vektörün iç çarpımı veya nokta çarpımı- **(x**.y) ile gösterilir:

$$\mathbf{(x,y)=x.}\,y=\mathbf{x}_1y_1+\mathbf{x}_2y_2+\mathbf{x}_3y_3+\ldots\ldots+\mathbf{x}_ny_n$$

dot(x,y) :x ve y **adlı (aynı-boyutta) iki vektörün (iç) nokta (aynı indisli elemanların) çarpımını** yapar ve sonucu bir **sayı**'ya atar.

```
>> x=[1 2 3];
>> y=[-1 2 1];
>> dot(x,y)
ans =     6
>> z=[1 2 3;4 5 6];
>> b=[1 -1 -3;7 0 -2];
```

```
>> dot(z,b)
ans =
29   -2   -21
```

8.1.3. Öklid (Euclidean) normu

Bir vektörün uzunluğu bu vektörün 'norm'u olarak adlandırılır. Öklid geometrisinde iki nokta arasındaki mesafe, her bir boyuttaki mesafelerin karelerinin toplamının karekökü alınarak hesaplanır.

MATLAB ortamında bir vektörün boyu, norm komutu ile hesaplanır.

norm (x): x vektörünün, (satır veya sütun) normunu hesaplar.

```
>> x=[1 2 3];
>> norm(x)
ans =
    3.7417
```

8.1.4. Üçgen eşitsizliği

Cauchy-Bernoulli-Schwarz eşitsizliği olarak da adlandırılan üçgen eşitsizliği, x ve y adlı iki vektörün iç çarpımlarının, bu iki vektörün ayrı ayrı normlarının çarpımından daha küçük veya eşit olduğunu gösterir. Bu eşitsizlik ancak a bir sayı olmak şartı ile y= ax olması halinde eşitlik haline dönüşebilir.

```
>> x=[1 2 3];
>> y=[-1 2 1];
>> abs(dot(x,y))
ans =    6
>> norm(x)*norm(y)
ans =   9.1652
```

8.1.5.Birim vektör

x adlı bir vektöre ilişkin birim vektör (u_x), x vektörü ile aynı yön ve doğrultuda olan fakat normu **1** olan vektördür. x vektörünün birim vektörü matlab ortamında;

```
>> x=[1 2 3];
>> ux=x/norm(x)
ux =
 0.2673    0.5345    0.8018
>> norm(ux)
ans =
 1
```

8.1.6. İki vektör arasındaki açı

x ve y adlı iki vektör arasındaki açı;

$\cos\theta = \frac{(x, y)}{\| x \| \| y \|}$ **olarak hesaplanır.**

```
>> x=[1 1 1];
>> xiz=[1 1 0];
>> teta=acos(dot(x,xiz)/(norm(x)*norm(xiz)))

teta =

   0.6155

>> tetader=(180/pi)*teta

tetader =

  35.2644
```

8.1.7. Ortogonallik (diklik)

Aralarındaki açı 90° (veya pi/ 2 radyan) olan iki vektörün ortogonal olduğu söylenir. Eğer x ve y ortogonal ise aralarındaki açı 90^0 olacağından $\theta = 90° \Rightarrow \cos\theta = 0$ yazılabilir.

$\cos\theta = \frac{(x, y)}{\| x \| \| y \|} = 0$ **(x,y)=0**

8.1.8. İzdüşüm

```
>> x=[1 2 3];
>> y=[-1 2 -3];
>> x=[1 2 3]';
>> y=[-1 2 -3]';
>> z=(dot(x,y)/norm(y)^2)*y

z = 0.4286
   -0.8571
    1.2857
```

8.2. Matrisler

‘m' satır, 'n' sütundan oluşan (m*n) boyutunda bir A matrisi;

$$A=\begin{bmatrix} a11 & a12 & . & a1n \\ a21 & . & . & . \\ . & . & . & . \\ am1 & . & . & . \end{bmatrix}$$

8.2.1. Matrisin evriği (transpozu)

Bir matrisin satırlarının sütun,sütunların satır yapılması, o matrisin **Evriğinin(Transpozu)** alınması işlemi olarak adlandırılır. B matrisi, A matrisinin evriği ise $B = A^T$ olarak ifade edilir. Bu işlem MATLAB ortamında B=A' şeklinde ifade edilir. Eğer A matrisi (m*n) boyutunda ise bu matrisin evriği olan B matrisi **(n*m)** boyutundadır.

```
>> A=[1 2 3; 4 -5 6]

A =
 1    2    3
   4   -5    6

>> B=A'
B =
   1    4
   2   -5
   3    6
```

8.2.2. Birim matris

Ana köşegen eleman değerleri 1, ana köşegen dışı elemanları 0 olan matris, birim matris olarak adlandırılır ve I ile gösterilir. Aşağıda (3*3) boyutunda bir birim (I) matris gösterilmiştir.

$$\mathbf{I}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Birim matris MATLAB ortamında eye komutu ile oluşturulur.

eye (n) : (n*n) boyutunda bir birim matris oluşturur.

eye(m,.n) :(m*n) boyutunda bir birim matris oluşturur. Ana köşegen elemanları T, köşegen dışı elemanlar ise 0 değerini alır.

```
>> A=eye(2)
A =
     1     0
     0     1
```

eye (s i z e (A)): A matrisi ile aynı boyutta bir birim matris oluşturur.

8.2.3. Matrisin **sayı** ile çarpımı

Bir matris bir sayı ile çarpılması, matrisin her bir elemanının o sayı ile çarpılması demektir.

```
>> C=[1 2 3;4 5 6;7 8 9];
>> D=2*C
D =
     2     4     6
     8    10    12
    14    16    18
```

8.2.4. Matrislerin toplanması ve çıkartılması

İki matrisin toplanabilmesi veya birbirlerinden çıkartılabilmesi için iki matrisin boyutlarının aynı olması gerekir. İki matris arasındaki işlemler (toplama-çıkarma) iki matrisin aynı indise sahip elemanları arasında gerçekleştirilir.

```
>> A=[1 2 3;4 5 6];
>> B=[0 -4 1;-1 -3 7];
>> C=A+B

C =

     1    -2     4
     3     2    13
```

8.2.5. İki matrisin çarpımı

İki matrisin çarpımının mümkün olabilmesi için birinci matrisin sütun sayısının ikinci matrisin satır sayışma eşit olması gerekir.

```
>> A=[2 5 1;0 3 -1]

A =
    2    5    1
    0    3   -1

>> B=[1 2 -1 4;5 -2 -3 3;0 -4 -5 6]

B =
    1    2   -1    4
    5   -2   -3    3
    0   -4   -5    6

>> C=A*B
C =
   27   -10   -22    29
   15    -2    -4     3
```

8.2.6. Matris tersinin hesaplanması

İki sayıdan biri diğerinin tersi ise bu iki sayının çarpımı **1** olmalıdır. Eğer birbirlerinin tersi olduğu söylenen sayı değil matris ise, bu iki matrisin çarpımı **(I)** birim matris olmalıdır.

$A*B = I \quad B = A^{1}$

A ve B matrisleri birbirlerinin tersi ise;

A*B=B*A=I

yazılabilir.

rank (A) : A matrisin rank'ını (bağımsız satır sayısını) bulur.

inv (A) : Eğer A matrisi tekil değilse A matrisinin tersini hesaplar.

```
>> A=[2,1;4,3]
A =
    2    1
    4    3
>> rank(A)
ans =
    2
>> B=inv(A)
B =
   1.5000   -0.5000
  -2.0000    1.0000
```

8.2.7. Matris kuvveti

MATLAB ortamında A matrisinin her bir elemanının k. dereceden kuvveti alınmak istendiğinde A.^k işlemi yapılmalıdır.

```
>> A=[1 2;3 4;5 6]

A =

     1     2
     3     4
     5     6

>> A.^2

ans =

     1     4
     9    16
    25    36
```

8.2.8. Matris determinantı

Bir matrisin determinantı bir sayıdır. Matris determinantını hesaplamak matris boyutu arttıkça zorlaşmaya başlar. MATLAB ortamında A matrisinin determinantı det(A) komutu kullanılarak hesaplanır. Eğer A matrisi kare matris değilse det(A) hesabı yapılamaz, hata mesajı ortaya çıkar.

```
>> A=[1 3 0;-1 5 2;1 2 1];
>> det(A)

ans =
      10
```

BÖLÜM 10

LİNEER DENKLEM SİSTEMLERİNİN ÇÖZÜMÜ

Sayısal hesaplamalarda lineer denklem sistemlerinin çözüm problemi ile çok sık karşılaşılır. Aşağıda bu tür problemlerin karşılaşıldığı bazı alanlar sıralanmıştır:

- Başlangıç değer yöntemleri yardımı ile iki noktalı sınır değer problemlerinin çözümünde,
- Sonlu fark yöntemleri ile iki noktalı sınır değer problemlerinin çözümlerinde,
- Sonlu fark teknikleri ile kısmi türevli diferansiyel denklemlerin çözümlerinde,
- Lineer programlama kullanılan optimizasyon tekniklerinde,
- İstatistik analizinde,
- En küçük kareler yaklaşımı ile eğri uydurulmasında,
- Lineerleştirme işleminden sonra bazı lineer olmayan denklem sistemlerinin çözümlerinde,
- Sonlu elemanlar yöntemi ile bazı mühendislik problemlerinin çözümünde

Genel olarak n. dereceden lineer denklem sistemi (açık formda);

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots & \\ \cdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \qquad (10.1)$$

şeklinde yazılır. Eğer lineer denklem sistemi matris formunda yazılır ise

$$Ax=b \qquad (10.2)$$

elde edilir. (10.2)'de görülen A matrisine 'katsayılar matrisi' denir. A matrisinin yapısı;

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{11} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \qquad (10.3)$$

x vektörü;

$$x = [x_1 \quad x_2 \quad \cdots \quad x_n]^T$$
(10.4)

ve ikinci taraf vektörü;

$$b = [b_1 \quad b_2 \quad \cdots \quad b_n]^T \qquad (10.5)$$

olur. Eğer b vektörü sıfır ise (10.2) ile verilen denklem sistemine 'homojen denklem sistemi' adı verilir. Bu durumda homojen sistemin geçerli çözümü, ancak n. dereceden A matrisinin rank'nın N'den küçük ise (rank(A) < N) mümkün olur. Bu durum aynı zamanda |a|=0 olması gerektiğini gösterir. Bu tür bir sistem (A*X=0 sistemi) |a|=0 olduğundan birden fazla çözüme sahiptir, b vektörünün sıfırdan farklı olması durumunda N bilinmeyenli lineer bir denklem sistemi oluşur. Genişletilmiş A matrisi;

C = [A|b]
(10.6)

olmak üzere, C matrisinin rankı ile A matrisinin rankının aynı olması durumunda (rank(C) = rank(A) ise) bu lineer denklem sistemi bir çözüme sahiptir. Bu durumda Ax=b sistemi bir yada daha fazla sayıda çözüm içerir. Homojen olmayan sistemin çözümünün tek olması için A katsayılar matrisinin rankı (bilinmeyen sayısı olan) n değerine eşit olmalıdır. Eğer rank(A)<N olursa sistemin belirli olmayan pek çok çözümü vardır.

10.1. Lineer denklem sistemlerinde çözüm yaklaşımları

Lineer denklem sistemlerinin çözümü için çeşitli yöntemler geliştirilmiştir. Bu yöntemler genel olarak direkt ve indirekt yöntemler olarak ikiye ayrılır;

A- Direkt yöntemler
Cramer yöntemi
Eliminasyon yöntemleri

Gauss eliminasyon yöntemi
Gauss-Jordan yöntemi
Aitken yöntemi
3-Yoğun eliminasyon yöntemleri
Doolitte yöntemi
LU ayrıştırma yöntemi
Crout yöntemi
Cholesky yöntemi
Banachiewicz yöntemi

4-Ortogonalleştirme yöntemleri
B- İndirekt yöntemler
Basit iterasyon (Jacobi iterasyonu) yöntemi
Gauss-Seidel iterasyon yöntemi
Relaxation yöntemi
Gradient yöntemi

Elektronik hesaplayıcıların ilk çıktığı yıllarda büyük boyutlu denklem sistemlerinin çözümünde yapılan yuvarlatma hataları ve bunların etkileri hakkında değişik görüşler ortaya atılmıştır. Hataların oluşumu ve etkileri eliminasyon yöntemleri ile araştırılmıştır. Yapılan analizler sonunda eliminasyon yöntemleri yuvarlatma hatalarına göre kararlıdır. Diğer yöntemlerin pek çoğu daha fazla sayıda işlem yapmayı gerektirirler ve kararlılıkları azdır. Çeşitli eliminasyon yöntemleri hakkında aşağıdaki ortak özelliklerden bahsedilebilir:

Ax=b denklem sisteminin çözümü sırasında A-1 matrisini hesaplamaya gerek yoktur.
Denklem sisteminin çözümünde Gauss-Jordan yöntemi diğer yöntemlere göre yavaş çalışır.
Bir denklem. sisteminin çözümüne A=LU şeklindeki çarpanlara ayırma işlemi ile başlama diğer yöntemlere göre en verimli yaklaşımdır.
A katsayılar matrisi 'pozitif tanımlı' ve simetrik ise, A=LU işlemi tavsiye edilir.

10.2. Gauss eliminasyon yöntemi

Bir matrisin satır ve sütunları arasında aşağıda belirtilen işlemlerin yapılması durumunda o matrisin değeri değişmez:

- Matrisin iki şatonun yer değiştirmesi,
- Bir satırın sıfırdan farklı bir sayı ile çarpılması,
- Bir satır bir sayı ile *çarpılır* ve başka bir satıra ilave edilir ve elde edilen yeni satır değeri üçüncü bir satır *yerine* yazılırsa.

- Gaussian eliminasyon yöntemi, katsayıları simetrik olmayan, içindeki sıfır sayısı nispeten az olan matris çözümünde daha verimli olarak kullanılır. Bu yöntemde A matrisine B matrisi ilave edilerek genişletilir ve elde edilen yeni (genişletilmiş) matrisin alt üçgeni (satır-sütun işlemleri kullanılarak forward-backward substution) sıfır yapılmaya çalışılır.
-
- Aşağıda verilen 3. dereceden lineer denklem sistemi dikkatlice incelenmelidir:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \Rightarrow \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}$$

$$\begin{bmatrix} 3 & 9 & 6 \\ 2 & 2 & 1 \\ -3 & -4 & -11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ -5 \end{bmatrix}$$

denklem sistemini saklayan x değerleri bulunmak istensin.

Adım 1: Öncelikle A matrisi genişletilmelidir:

$$\left[\begin{array}{ccc|c} 3 & 9 & 6 & 3 \\ 2 & 2 & 1 & 4 \\ -3 & -4 & -11 & -5 \end{array}\right]$$

Adım 2: 1. satır dışındaki tüm satırlardaki X1 katsayıları sıfırlanmalıdır. Elde edilen genişletilmiş A matrisinin ilk satırını olduğu gibi bırak.

İlk işlem 2. satıra uygulanmalıdır; İlk satırı a21 =2 ile çarp a11 = 3'e böl ve elde edilen yeni satırı eski 2. satırdan çıkart. Elde edilen satırı 2. satır olarak ata:

$$\begin{array}{l} a_{11}x_1+a_{12}x_2+a_{13}x_3=b_1 \\ \quad a_{22}'x_2+a_{23}'x_3=b_2 \;\Rightarrow \\ a_{31}x_1+a_{32}x_2+a_{33}x_3=b_3 \end{array} \qquad \left[\begin{array}{cccc} 3 & 9 & 6 & \mathbf{3} \\ 0 & -4 & -3 & \mathbf{2} \\ -3 & -4 & -11 & \mathbf{-5} \end{array}\right]$$

Benzer işlem 3. satıra uygulanmalıdır; İlk satırı a31 =-3 ile çarp a11 = 3'e böl ve elde edilen yeni satırı eski 3. satırdan çıkart. Elde edilen satırı 3. satır olarak ata:

$$\begin{array}{l} a_{11}x_1+\ a_{12}x_2+\ a_{13}x_3=b_1 \\ \qquad a_{22}'x_2+\ a_{23}'x_3=b_2' \Rightarrow \\ \qquad a_{32}'x_2+\ a_{33}'x_3=b_3' \end{array} \left[\begin{array}{ccc|c} 3 & 9 & 6 & 3 \\ 0 & -4 & -3 & 2 \\ 0 & 5 & -5 & -2 \end{array}\right]$$

Adım 3: 2. satır dışında ileriye doğru tüm satırlardaki x2 katsayıları sıfırlanmalıdır. Amaç ana köşegenin alt tarafını tümüyle sıfırlamaktır:
İkinci satırı a32 = 5 ile çarp a22 = -4 'e böl ve elde edilen yeni satırı eski 3. satırdan çıkart. Elde edilen satırı 3. satır olarak ata:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{22}'x_2 + a_{23}'x_3 &= b_2' \\ a_{33}''x_3 &= b_3'' \end{aligned} \Rightarrow \begin{bmatrix} 3 & 9 & 6 & 3 \\ 0 & -4 & -3 & 2 \\ 0 & 0 & -35/4 & 0.5 \end{bmatrix}$$

Son durum için eşitlikler tekrar yazılır ise;

$$\begin{aligned} 3x_1 + 9x_2 + 6x_3 &= 3 \\ -4x_2 - 3x_3 &= 2 \\ (-35/4)x_3 &= 0.5 \end{aligned}$$

elde edilir. Yerine koyma yaklaşımı kullanarak sıra ile x3 =-0.0571, x2 =-0.4571 ve x1 =2.4857 değerleri bulunur.

10.2.1. Gauss eliminasyon yönteminin tuzakları

Basit Gauss eliminasyon yöntemi ile çözülebilecek bir çok denklem sistemi olmasına karşın, yöntemi uygulamak için genel bir bilgisayar programı yazmadan önce araştırılması gereken bazı tuzak noktalar vardır. Bu tuzaklar (diğer eliminasyon yöntemleri için de geçerlidir) aşağıda maddeler halinde açıklanmıştır:

- Hem ileriye doğru eleme hem de geriye doğru yerine koyma aşamalarında sıfıra bölme olasılığıdır. A matrisinin katsayılarından birinin sıfıra yakın olması durumunda da sorun ortaya çıkar. Bu sorunları kısmen çözebilmek için pivotlama tekniği geliştirilmiştir.

- Bilgisayarlar yaptıkları çarpma, bölme vb. işlemlerde hesaplama sonunda elde edilen rakam değerinde yuvarlama yapar ve bu yeni (yuvarlatılmış) değeri bir sonraki işlemde kullanır. Özellikle büyük sayıda denklem çözüleceği zaman yuvarlama hataları (daha sonraki hesaplama adımlarında) büyük önem kazanır. Genel olarak 100 adımlık işlemlerdeki yuvarlama hataları önem kazanmaya başlar. Hesaplama sonunda elde edilen değerleri Ax=b lineer eşitliğinde yerlerine koyarak bariz bir hatanın oluşup oluşmadığı kontrol edilebilir.

Ax = b eşitliğinde yer alan A matrisi 'hasta' matris olabilir. Aşağıda 'hasta' matris tanımına uyan bir Örnek matris gösterilmektedir:

```
» A=[3.021 2.714 6.913;1.031 -4.273 1.121;5.084 -5.832 9.155]
A =
     3.0210    2.7140    6.9130
     1,0310   -4.2730    1.1210
     5.0840   -5.8320    9.1550
»b=[12.648;-2.121;8,407]              ('enter')
b =
   12.6480
   -2.1210
    8.4070
»A\b                                              ('enter')
ans =
```

```
 1.0000
 1.0000
 1.0000
```

Eğer A matrisindeki (2,2) elemanının değeri çok az değiştirilir ise (-4.2730 yerine -4.2750) kullanılırsa);

```
» A(2,2)= - 4.2750     ('enter')


A=
    3.0210   2.7140    6.9130
    1.0310   -4.2750    1.1210
    5.0840   -5.8320    9.1550
»A\b    ('enter')
ans =
   -1.7403
    0.6851
    2.3212
```

elde edilir. Sonuçtan da görüldüğü gibi A matrisindeki çok küçük bir değişim x matrisinin değerini çok fazla değiştirmiştir. Bu nedenle A matrisi 'hasta' matris olarak adlandırılır. Bunun nedeni A matrisini oluşturan yüzeyden en az ikisinin birbirlerine paralel olma durumuna gelebilmelerinin bu iki yüzeyin arakesit noktasının çok hassas olmasından kaynaklanmaktadır. Arakesit noktasındaki küçük bir değişme yüzeylerin eğimlerini ciddi olarak etkilemektedir. MATLAB ortamında A matrisinin 'hastalık' derecesini Ölçen cond komutu bulunmaktadır. Bu komut hastalık seviyesi ölçülecek bir matrise uygulandığında sonuç ne kadar büyük olursa 'hastalık' o kadar tehlikeli demektir. En iyi değer 1 dir (matrisin hasta olmadığım gösterir). Örneğin birim matris (I) için cond (I} değeri 1 olarak hesaplanır. Yukarıda verilen A matrisine bu komut uygulandığında;

```
» cond (A)
ans =
    1.7658e+016
```

elde edilir. Görüldüğü gibi A matrisi ileri derecede hasta bir matristir. Yuvarlama hatalarının 'hasta' sistemler için ne kadar önemli olduğu ortadadır. Hasta sistemlerde, yuvarlatma hataları nedeni ile, hesaplama sonunda elde edilen x değerleri gerçek sistem eşitliklerini (Ax=b) genellikle sağlamazlar. Mühendislik problemlerinden elde edilen lineer denklemlerin çoğunda, doğaları gereği, A matrisi 'hasta' matris özelliklerine sahip değildir.

- Eğer (N*N) boyutlu A matrisi içinde birbirlerinin aynısı (yada katları) olan iki satır (yada sütun) mevcut ise A matrisinin rankı 1 azalır (N-3 olur), bu durumda N adet bilinmeyeni çözmek için N-l adet denklem kullanılır. Bu durumda çözüm imkânsızdır. Bu durumda A matrisi 'tekil' olacaktır ve daha önce de bahsedildiği gibi tekil matrislerin determinantları sıfırdır. Eliminasyon işlemleri yapılırken genişletilmiş A matrisinin köşegen elemanlarından bir tanesi sıfır olduğunda algoritma içine bir kesme komutu konularak çıkışa da matrisin 'tekil' olduğu ve bu nedenle işlemin devam edemeyeceği uyarısı yazılabilir.

10.2.2. Eliminasyon yöntemlerinin tuzaklarını giderme

Eliminasyon yöntemlerinin tuzaklarını gidermek için başlıca üç yaklaşım kullanılır:

- Hesaplamalarda daha fazla anlamlı basamak kullanmak. Sayıların duyarlılıklarını artırmak.

- Gauss eliminasyon yönteminde ileri doğru hesaplamalarda paydaya getirilen (pivot) sayı sıfır olduğunda sorun çıkar. Eğer pivot eleman sıfır olmayıp sıfıra yakın bir sayı olsa dahi sorun ortadan kalkmış olmaz. Bu durumda pivot elemanı diğer matris elemanlarına oranla küçük olursa yuvarlama hataları ortaya çıkacaktır. Pivot elemanının altındaki sütunun en büyük katsayısını belirlemek bir avantaj sağlar. Satırların yeri, en büyük eleman pivot elemanı olacak şekilde değiştirilebilir. Bu işleme 'kısmi pivotlama' adı verilir. Eğer sütunlarla birlikte satırlarda en büyük eleman için taranırsa ve sonra yer değiştirilirse bu sürece 'tam pivotlama' adı verilir. Tam pivotlama ortaya çıkan karışıklıklar nedeni ile sık kullanılmaz. Sıfıra bölmeyi önlemenin yanında, pivotlama aynı zamanda yuvarlama hatalarını da en aza indirir. Böylece 'hasta' sistemlere karşı bir tedbir alınmış olur.

Aşağıda 'kısmi pivotlama' yaparak Ax=b lineer matris eşitliğim çözen bir MATLAB programı verilmiştir.

```
%   GAUSS ELİMİNASYON YÖNTEMİ İLE Ax=b LİNEER EŞİTLİĞİNİN ÇÖZÜLMESİ
%   pivotlama kullanılıyor
%   A matrisi N*N boyutunda tekil olmayan
%   katsayılar matrisidir. b matrisi N*l boyutundadır
% x matrisi N*l boyutunda çözüm matrisidir
% x matrisinin eleman değerleri başlangıçta sıfır alınmaktadır
% C matrisi program içinde geçici olarak kullanılmaktadır
 A=[l 2 1 4;2 0 4 ;422 l;-3 1 3 2]; b=[13; 28; 20;6];
 [N N]=size(A);
 x=zeros(N,1);
 C=zeros(1,N+1);
% Aşağıda genişletilmiş [A|b] matrisi oluşturulmaktadır
genis=[A b];
for p=l:N-l;
      % p kolonu için kismi pivotlama yapiliyor
    [Y,J]=max(abs(geniş <p:N,p)));
      % p ve J kolonlarinin yerleri değiştiriyor
    C=genis(p, :};
     geniş(p,:)=genis(J+p-l,:};
    geniş(J+p-1,:)=C;
 if geniş(p,p)==0
    disp('A matrisi tekil olduğu için program durduruluyor'}
    break
  end
    % p. kolon için eliminasyon başliyor
```

```
    for k=p+1:N
        m=genis(k,p)/geniş(p,p);
        geniş(k,p:N+l)=genis(k,p:N+l)-m*genis(p,p:N+l);
  end
end
% yerine koyma işlemi basliyor
A=genis(1:N,1:N);
b=genis(1:N,N+1);
x(N)=b(N)/A(N,N);
       for k=N-l:-l:l
           x(k)=(b(k)-A(k,k+l:N)*x(k+l:N))/A(k,k);
       end
x
```

Aşağıda, yukarıda verilen MATLAB programının uygulanması sonunda elde edilen çözüm matrisi
görülmektedir:

```
x =
      3
     -1
      4
      2
```

- Genişletilmiş A matrisinin bir satırındaki katsayılar ile diğer bir satırının katsayıları arasında büyük farklılıklar olması durumunda yuvarlama hatalarının en aza indirmek için ölçekleme yapılabilir. Her bir eşitlikteki en büyük katsayı T olacak şekilde katsayılar arasında işlem yapılması ölçekleme olarak adlandırılır. Daha sonra ise her bir satırdaki katsayılar en büyük değeri katsayılar matrisinde köşegene getirmek için pivotlama yapılmalıdır. Ölçekleme yuvarlatma hatalarım en aza indirmek için kullanılmasına rağmen ölçeklemenin kendisi de yuvarlatma hatası meydana getirebilir. Bu nedenle ölçeklenmiş değerler, sadece pivotlamaya bir kriter olması amacı ile hesaplanmalı, eleme ve yerine koyma işlemleri için orijinal denklem katsayıları saklanmalıdır.

10.3. Gauss-Jordan eliminasyon yöntemi

Gauss-Jordan yöntemi Gauss yönteminin bir başka şeklidir. İki yöntem arasındaki temel fark; Gauss-Jordan yönteminde bir bilinmeyen elendiğinde, sadece izleyen satırlarda değil bütün denklemlerden elenmesidir. Ayrıca tüm satırlar pivot elemanlara bölünerek normalize edilirler. Böylece eliminasyon aşaması sonunda üçgen matris yerine birim matris elde edilir. Bu nedenle çözüme ulaşmak için yerine koyma (back substitution) işlemine gerek kalmaz. Gauss yöntemi için geçerli olan tüm tuzaklar Gauss-Jordan yöntemi içinde geçerlidir. Gauss-Jordan yöntemi Gauss yöntemine göre daha fazla hesaplama (yaklaşık %50 daha fazla) gerektirdiği için kullanıcı açısından pek tercih edilmez.

10.4. LU ayrıştırma yöntemi

Gauss eliminasyon yöntemi, Ax=b lineer eşitliğinde A ve b matris elemanlarının birbirlerine yakın değerler aldığı durumlarda çok verimli bir yöntem olmasına karşın, b matris elemanlarının A matris elemanlarından (orantısız anlamında) farklı değerler aldığı durumlarda verimsiz hale gelmektedir. Bilindiği gibi Gauss yönteminde ileri ve geri (yerine koyma) işlemleri yapılmaktadır. İleriye doğru olan hesaplamalar oldukça büyük zaman almaktadır.

LU (L-Lower, U-Upper) ayrıştırma yöntemi zaman alıcı A matrisinin elenmesini sağ taraf (b matrisi) işlemlerinden ayırır. Böylece A matrisi bir kere ayrıştırıldığı zaman birden fazla sağ taraf vektörü verimli bir şekilde hesaplanabilir. LU yaklaşımında iki adım söz konusudur;

- LU ayrıştırma adımı; A matrisi, alt L ve üst U adlı üçgen matrislerin çarpımı şekline getirilir.
- Yerine koyma adımı; b sağ taraf matrisi için x çözümünü belirlemek için L ve U matrisleri kullanılır.

A=LU

Matris eşitliğinde, L matrisi alt üçgen matris (ana köşegenin altında kalan elemanlar sıfırdan farklı), U ise üst üçgen (ana köşegenin üstünde kalan elemanlar sıfırdan farklı) matris olarak adlandırılır. A matrisi reel veya kompleks olabilir. LU ayrıştırma yöntemi Gauss yöntemine göre biraz daha iyidir, LU yaklaşımı adımlar halinde aşağıda açıklanmıştır:

Ax=b;LU=A

LUx=b (B matrisi sütun vektörü olmayabilir)

L matrisi alt üçgen matris olduğundan ileriye doğru (satırlar arası) işlemlerde verimli olur:

Ly=b; y=Ux

x değerini bulmak için,

Ux=y

eşitliği çözülür. U matrisi üst üçgen matris olduğundan geri (yerine koyma) işlemlerinde Ux=y denkleminin çözümü daha verimli olacaktır.

Yukarıda özelliği açılanan LU yaklaşımı,

$$\begin{bmatrix} 4 & 1 & 1 \\ 2 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 3 \\ 7 \end{bmatrix}$$

eşitliğine adım adım uygulansın:

$$A=LU \Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 4 & 1 & 1 \\ 2 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

121*2/4, i_{31} =2/4,

u22=-l~(İ/2)(l) = -1.5, u23-l-(l/2)(l) = 0.5,

$$4 = a_{11} = u_{11}, \quad 1 = a_{12} = u_{12}, \quad 1 = a_{13} = u_{13},$$
$$2 = a_{21} = l_{21}u_{11}, \quad -1 = a_{22} = l_{21}u_{12} + u_{22}, \quad 1 = a_{23} = l_{21}u_{13} + u_{23}$$
$$2 = a_{31} = l_{31}u_{11}, \quad 1 = a_{32} = l_{31}u_{12} + l_{32}u_{22}, \quad 1 = a_{33} = l_{31}u_{13} + l_{32}u_{23} + u_{33}$$

$$l_{21} = 2/4, \; l_{31} = 2/4,$$
$$u_{22} = -1 - (1/2)(1) = -1.5, \; u_{23} = 1 - (1/2)(1) = 0.5,$$
$$l_{32} = \frac{1}{-1.5}(1 - (1/2)(1) = -1/3,$$
$$u_{33} = 1 - (1/2)(1) - (-1/3)(1/2) = 2/3$$

Yukarıda elde edilen sonuçlar kullanılarak;

$$\begin{bmatrix} 4 & 1 & 1 \\ 2 & -1 & 1 \\ 2 & 1 & 1 \end{bmatrix} = LU = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & -1/3 & 1 \end{bmatrix} \begin{bmatrix} 4 & 1 & 1 \\ 0 & -3/2 & 0.5 \\ 0 & 0 & 2/3 \end{bmatrix}$$

bulunur. Elde edilen eşitlik Ly=b olarak kabul edilirse;

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & -1/3 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 3 \\ 7 \end{bmatrix}$$

bulunur. Matris çarpımından,

y_1=9

y2 =3-0.5y1 =-1.5

y_3=7-0.5y_{1+}(l/3)y_2=2

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 9 \\ -1.5 \\ 2 \end{bmatrix}$$

elde edilir. Ux=y eşitliği kullanılarak;

$$\begin{bmatrix} 4 & 1 & 1 \\ 0 & -3/2 & 0.5 \\ 0 & 0 & 2/3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ -1.5 \\ 2 \end{bmatrix}$$

bulunur. Yerine koyma yöntemi kullanılarak (back substitution);

x3=3

x2=-(2/3)(-l.5-0.5x3) = 2

$X_1=(1/4)(9 -x_2-x_3) = l$

elde edilir. A matrisinin LU ayrıştırma yöntemi ile iki çarpım matrisine ayrıştırılması için

MATLAB ortamında lu komutu ile kullanılır:

[LA, UA] = lu (A): A matrisini LU yöntemi kullanarak LU ve UA adlarında iki matrisin çarpımı haline dönüştürür." Kullanıcı iki matrisi MATLAB ortamında LA ve LU olarak adlandırmak zorunda değildir.

P permütasyon matrisinin hesaplanmasında MATLAB ortamında aşağıda verilen komut türü kullanılır:

[L1A, UA, P] = lu(A) : Bu komut türünde P; permütasyon matrisi olarak adlandırılır ve dört matris arasında;

L1A*UA=P*A veya P'*L1A*UA=A; (P'*L1A=LA) ilişkisi vardır.

Aşağıdaki örnek bu komut yardımı ile çözülebilir:

»A=[l 2 6; 4 8 1; -2 3 5] (‘enter’)

A =

```
 1     2     6
 4     8    -1
-2     3     5
```

»[L1A, UA, P]= lu(A) (‘enter’)

L1A =

```
 1.0000     0     0
-0.5000  1.0000    0
 0.2500     0  1.0000
```

UA =

```
4.0000  8.0000  -1.0000
     0  7.0000   4.5000
     0       0   6.2500
```

```
P =
     0     1     0
     0     0     1
     1     0     0
```

elde edilir. Görüldüğü gibi LİA matrisinin üst üçgeni ve UA matrisinin alt üçgeni tamamen sıfırlardan oluşmaktadır.

Önemli not: '\’ işareti MATLAB ortamında Gauss ayrıştırma yöntemim ifade etmek için kullanılır.

```
% Ax=b lineer matris eşitliğinin *\' komutu ile çözülmesi
%
»A=[1  2  6;4  8  -l;-2  3  5];     ('enter')
» b=[15; -20; 54];                  ('enter')
» x=A\b                            ('enter')
x=
-12.6571
 4.2286
         3.2000
```

bulunur. Görüldüğü gibi elde edilen x değeri LU ayrıştırma yönteminde elde edilen X değeri ile aynıdır.

Ax-b
eşitliği yerine
xTAT =bT
eşitliği kullanılırsa, sağdan bölme yöntemi ile;
xT=bT/AT

elde edilir. Yukarıdaki işlem MATLAB ortamında;

```
» A =  [4    1    1; 2    -1    1;  2    1    1];          ('enter) ('enter')
 » b= [9; 3; 7];                                          ('enter')
%   At matrisi A matrisinin devriği olsun
 » At =A'
%   A matrisinin devriği
At =
        4        2        2
        1        -1       1
        1        1        1


% bt vektörü b vektörünün devriği olsun
 bt =
             9   3              7
            10   5              -1

% x vektörünün devriği
 » xt=bt /At                                              ('enter')
xt =
     12     3
```

elde edilir. Görüldüğü gibi yukarıdaki sonuç ile aynı değerler elde edilmiştir. Aşağıda verilen lineer denklem sistemini sağlayan [x y z w] satır vektörünün değerleri Gauss ayrıştırma metodu (MATLAB komutunu) kullanarak bulunabilir:

10.5. Doğrusal eşitliklerin çözümünde matris tersinin kullanılması

Ax=b

matris eşitliği soldan A -1 ile çarpılırsa;

A-1Ax = A-1b

Ix = A-1b (I; birim matris)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = A^{-1}b$$

elde edilir. Yukarıda verilen matris eşitliği MATLAB ortamında çözülebilir:

```
» A - [3 2 -1; -1 3 2; 1 -1 -1]                    ('enter')
A=
        3       2       -1
        -1      3        2
        1       -1      -1
                                                    (‘enter')

 » b= [10; 5; -1]                                   ('enter')
 b =
        10
         5
        -1

» x = inv(A) *b                                     ('enter')

x =
        -2.0000
         5.0000
        -6.0000

% alttaki satir sonucu test etmek için konulmuştur
 » A*x                                              ('enter')

ans =
        10,0000
         5.0000
        -1.0000
```

10.6. Basit (Jacobi) iterasyon yöntemi

Genel olarak itératif yöntemlerde izlenen yol; denklem çözümüne yaklaşık bir çözüm takımı ile başlamak ve belirli bir algoritmayı tekrarlayarak, gerçek çözümü en az hata ile hesaplamak mantığına dayalıdır. Dolaylı yöntemler olarak da adlandırılan bu yöntemler, hem algoritmalarının kolayca hesaplanabilir olmaları, hem do yuvarlama hatalarının en az ve iterasyon sayısı arttıkça hata birikimi olmaması açısından sık kulladırlar. Basit (diğer adı ile Jacobi) iterasyon adı verilen yaklaşım ile Gauss-Seidel iterasyon yöntemi itératif yöntemlerin başında gelmektedir. Ancak unutulmamalıdır ki itératif yöntemlerin en önemli problemi, yakınsama problemidir. Bazı tip

problemlerde Gauss-Seidel, bazı tip problemlerde ise Jacobi iterasyonu daha çabuk yakınsar. Eğer bir problemde her iki itératif yaklaşım kullanılarak yakınsama sağlanıyor ise daha hızlı olması nedeni ile 'Gauss-Seidel' yöntemi tercih edilmelidir.

Jacobi iterasyonunun yakınsayabilmesi için A matrisinin ana köşegen üzerindeki elemanlarının mutlak değerlerinin bir koşulu (strictly diagonally dominant) yerine getirmesi gerekmektedir; A matrisinin i. satırının köşegen üzerindeki değeri (aii)'nin mutlak değeri A matrisinin i. satırındaki tüm elemanların mutlak değerlerinin toplamından büyük olmalıdır.

Aşağıda verilen lineer denklem sistemi Jacobi iterasyonu ile çözülsün:

$$\begin{aligned} 4x_1 - 2x_2 + x_3 &= 6 \\ 4x_1 - 8x_2 + x_3 &= -20 \\ -x_1 + x_2 + 5x_3 &= 11 \end{aligned} \Rightarrow \begin{bmatrix} 4 & -2 & 1 \\ 4 & -8 & 1 \\ -1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -20 \\ 11 \end{bmatrix} \Rightarrow Ax = b$$

Yukarıdaki eşitliklerden;

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

elde edilir. Eğer program $x_1^{(0)} = 1$; $x_2^{(0)} = 2$; $x_3^{(0)} = 1$ başlangıç değerleri ile başlar ise;

$$x_1^{(1)} = \frac{6 + 2*2 - 1}{4} = 2.25; \quad x_2^{(1)} = \frac{20 + 4*1 + 1}{8} = 3.125; \quad x_3^{(1)} = \frac{11 + 1 - 2}{5} = 2$$

birinci iterasyon sonunda bulunan değerler ile ikinci iterasyona gidilir ve bu işlem bu şekilde devam ettirilir ise Tablo 10.1'de gösterildiği gibi 15. iterasyonda (verilen tolerans için);

$x_1^{(15)} = 3.1745$; $x_2^{(15)} = 4.3333$; $x_3^{(15)} = 1.9683$

Tablo 10.1

Iterasyon sayısı	x_1	x_2	x_3
1	2.2500	3.1250	2.0000
2	2.5625	3.8750	2.0250
3	2.9312	4.0344	1.9375
4	3.0328	4.2078	1.9794
5	3.1091	4.2638	1.9650
6	3.1407	4.3002	1.9690
7	3.1578	4.3165	1.9681
8	3.1662	4.3249	1.9683
9	3.1704	4.4.3291	1.9683
10	3.1725	4.3312	1.9683
11	3.1736	4.3323	1.9683
12	3.1741	4.3328	1.9683
13	3.1743	4.3331	1.9683
14	3.1745	4.3332	1.9683
15	3.1745	4.3333	1.9683

değerleri x çözüm matrisini belirler. İterasyonu durdurmak için kullanılabilecek bir kriter j. iterasyonun
sonunda elde edilen xj çözüm matrisi ile bundan bir önceki (j-1) iterasyonda elde edilen x(j-1) çözüm matrisi arasındaki farkın mutlak değeri olarak başlangıçta belirlenen epsilon değerinden küçük olmasıdır. Buna benzer bir çok durdurma kriteri geliştirilebilir.

İteratif yaklaşımlar için diğer Önemli bir nokta da x(0) başlangıç değerlerinin abartılı olarak seçilmesi durumunda yakınsamanın gecikebileceği gerçeğidir. İteratif yöntemler için verilen sisteme ilişkin uygun değerler (daha önceden) elde edilebilmiş ise bu değerlerin kullanılması yakınsamanın kolaylaşması açısından çok uygun olur.

Verilen eşitliklerin yakınsaması beklenmekteydi zira A matrisinin köşegen elemanları daha önce belirtilen özellikleri sağlamaktaydı;
$|4| > |-2| + |1|$
$|-8| > |4| + |1|$
$|5| > |-1| + |1|$

Aşağıda MATLAB programlama dilinde yazılmış ve Jacobı iterasyonu ile lineer denklem çözümünde kullanılan program verilmiştir;

Not : eps =2.2204e-016 değerinde bir MATLAB komutudur.

% Ax=b lineer matris eşitliğinin JACOBI iterasyonu ile çözümü

```
% A; matrisi N*N boyutunda tekil olmayan katsayilar matrisidir.
% b; matrisi N*l boyutundadir
% x; matrisi 1*N boyutunda çözüm matrisinin evriğidir
% p; matrisi N*l boyutunda baslangic değerler matrisidir
% delta; iterasyorıun son iki adimi arasindaki müsade edilebilen
% fark değeridir
% maxiter; maksimum iterasyon sayisìdir.Eger kullanìcì maxiter
% sayisina kadar iterasyon yakinsamaz ise programi durdurmak için
% bu değeri kullanir
%
A=[4 -2 1;4 -8 1;-1 1 5]; b=[6;-20;11]; p=[1;2;1];
N=length(b);
x=zeros(1,N);
maxiter=30; delta=0.00001;
for k=l:maxiter
    for J=1:N
      x(J)=(b(J)-A(J,[1:J-1,J+1:N])*p([1:J-1,J+1:N]))/A(J,J);
     end
     hata=abs(norm(X'-p));
    hatatek=hata/(norm(X)+eps);
     p=x' ;
    if (hata<delta)|(hatatek<delta)
        disp(' iterasyon maxiter den önce sona erdi')
        break
     end
end
display('iterasyon sayisi=');
display (k-1) ;
x=x'
```

Yukarıda verilen programın uygulanması sonunda aşağıdaki sonuçlar elde edilir:

```
»
iterasyon maxiter den önce sona erdi
iterasyon sayisi=
                          15
x =
         3.1746
         4.3333
         1.9683
```

10.7. Gauss-Seidel iterasyon yöntemi

Jacobi iterasyonunda $X^{(0)}$ başlangıç değerleri sırayla;

eşitliklerinde yerlerine konulmuştu. Gauss-Seidel yönteminde ise ilk eşitlikte (X_1' e ilişkin) $X_2^{(0)}$ =2; $X_3^{(0)}$ =1 değerleri yerlerine konur. Elde edilen $X_1^{(1)}$ = 2.25 değeri (yine aynı iterasyon içindeki);

$$x_2^{(1)} = \frac{20 + 4x_1^{(1)} + x_3^{(0)}}{8} = \frac{20 + 4 * 2.25 + 1}{8} = 3.75$$

eşitliğinde yerine konulur. Yukarıda elde edilen $X_1^{(1)}$ ve $X_2^{(1)}$ değerleri ise $x_3^{(1)}$ eşitliğinde yerine konulursa;

$$x_3^{(1)} = \frac{11 + x_1^{(1)} - x_2^{(1)}}{5} = 1.9$$

bulunur. Aynı problem Jacobi iterasyonu ile çözüldüğünde ilk iterasyon sonunda;

$$x_1^{(1)} = 2.25 \; ; \; x_2^{(1)} = 3.125 \; ; \; x_3^{(1)} = 2$$

bulunmuştu. Gauss-Seidel yaklaşımında ise

$$x_1^{(1)} = 2.25 \; ; \; x_2^{(1)} = 3.75 \; ; \; x_3^{(1)} = 1.9$$

bulunur. Böyle bir yaklaşım yakınsamayı çabuklaştırarak hesaplamanın daha çabuk bitmesini temin eder. Gauss-Seidel yaklaşımında da Jacobi yaklaşımında olduğu gibi, A matrisinin ana köşegen elemanlarının mutlak değerleri, aynı satır üzerinde yer alan diğer elemanların mutlak değerlerinin toplamından daha büyük olmalıdır. Aksi halde yakınsama sağlanamaz.

Tablo 10.2

Iterasyon sayısı	X_1	x_2	x_3
1	2.2500	3.7500	1.9000
2	2.9000	4.1875	1.9425
3	3.1081	4.2969	1.9622
4	3.1579	4.3242	1.9667
5	3.1704	4.3311	1.9679
6	3.1736	4.3328	1.9682
7	3.1743	4.3332	1.9682
8	3.1745	4.3333	1.9682

Yukarıdaki işlem diğer iterasyonlar içinde yapıldığında elde edilen sonuçlar Tablo 10.2'de gösterilmiştir. Tablo 10.1 ile Tablo 10.2 arasında bir karşılaştırma yapıldığında Gauss-Seidel yaklaşımında çok daha az iterasyon sayısı ile yakınsama sağlandığı görülmektedir. Her iki yaklaşımın aynı probleme aynı ilk koşullar altında uygulandığı da unutulmamalıdır.

MATLAB programlama dilinde yazılmış ve Gauss-Seidel iterasyonu ile lineer denklem sistemi çözümünde kullanılan program aşağıda verilmiştir:

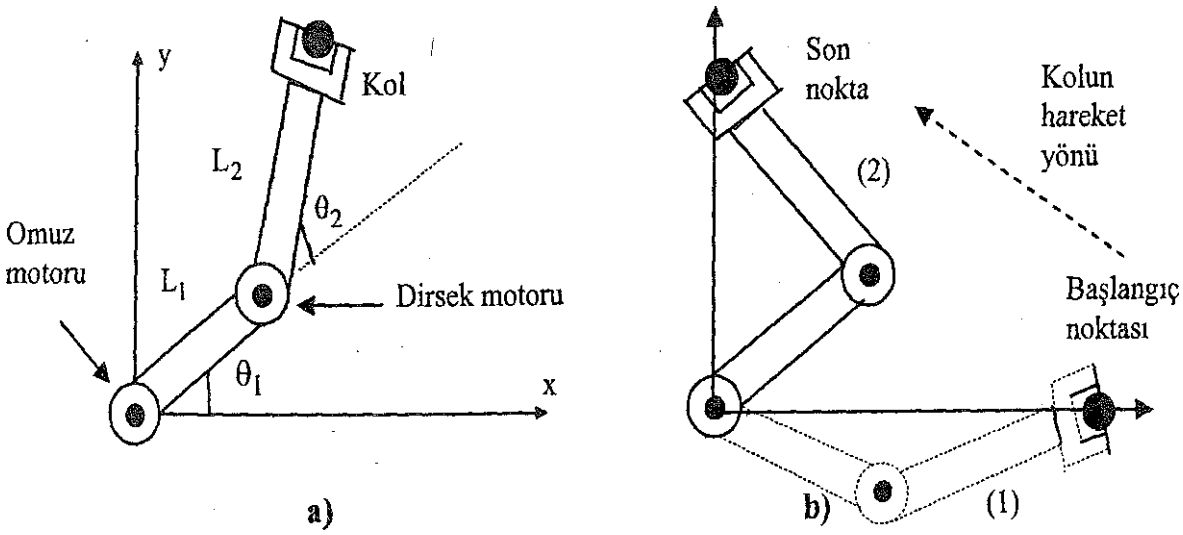
```
% Ax=b lineer matris eşitliğinin Gauss-Seidel % iterasyonu ile çözümü
% x; matrisi 1*N boyutunda çözüm matrisidir
% p; matrisi N*l boyutunda baslangic değerler matrisidir
% delta; iterasyonun son iki adimi arasindaki müsade edilebilen
% fark değeridir
% maxiter; maksimum iterasyon sayisidir.
%
% Eger maxiter sayisina kadar iterasyon
% yakinsamaz ise programi durdurmak için bu değer kullanilir

A=[4 -2 1;4 -8 1;-1 1 5];
b=[6;-20;11];
p=[l;2;l];
N=length(b);
x=zeros(1,N);
maxiter=30; delta=0.00001;
for k=1:maxiter
      for J=1:N
        if J==l
       x(l)=(b(l)-A(l,2:N)*p(2:N))/A(1,1);
       elseif J==N
       x(N)=(b(N)-A(N,1:N-1)*(x(l:N-l))')/A(N,N);
       else
       X(J) = (b(J)-A(J,l:J-l)*x(l:J-l)-A(J,J+l:N)*p(J+l:N))/A(J,J) ;
       end
     end
     hata=abs (norm(x'-p) ) ;
     hatatek=hata/(norm(x)+eps);
     p=x' ;
   if (hata<delta)|(hatatek<delta)
        disp('iterasyon maxiter den önce sona erdi ‘)
        break
    end
    x
end
display(‘iterasyon sayisi=’);
display(k-1) ;
x=x!
```

Not: eps =2.2204e–016 değerinde bir MATLAB komumdur.

Yukarıda verilen programın uygulanması sonunda aşağıdaki sonuçlar elde edilir:

```
»
    iterasyon maxiter den önce sona erdi
    iterasyon sayisi=

ans =
8
X     =
3.1746
4.3333
1.9683
```

10.8. Bir uygulama olarak robot kontrolü



Şekil 10.1

Şekil 10.1'de robot kol mekanizması gösterilmiştir. Tüm sistem iki adet motor tarafından tahrik edilmektedir. Motorlardan biri 'omuz motoru' olup θ_1 açısını kontrol etmekte, diğer motor ise 'dirsek motoru' olarak adlandırılmış olup θ_2 açısını kontrol etmektedir. Kolun ucunda görülen 'el'in koordinatları;

$$x = L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2)$$

$$y = L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

olarak verilmektedir. Bu hareket probleminde ana sorun, 'el'in bir noktadan diğer bir noktaya olan hareketini sağlamak için (iki adet) eklem motorlarının konum (θ1,θ2) açılarının nasıl kontrol edileceğidir. Şekil 10.1 (b)'de gösterildiği gibi 'kol' (1) numaralı konumdan (2) numaralı konuma doğru hareket edecektir. Kol, (1) pozisyonunda duruyorken daha sonra hızlanmalı ve (2) numaralı pozisyonda hızı tekrar sıfıra düşmelidir.

Motor kontrolörlerine gönderilecek açı bilgilerine ilişkin polinom ifadeleri;

$\theta_1(t) = \theta_1(0) + a_1t^5+a_2t^4+a_3t^3+a_4t^2+a_5t$

$\theta_2(t) = \theta_2(0) + b_1t^5 + b_2t^4 +b_3t^3 +b_4t^2 +b_5t$

olarak verilmektedir (polinomun neden 5.dereceden olduğu aşağıda açıklanacaktır). Yukarıda görülen $\theta_1(0)$, $\theta_2(0)$ açıları L_1 ve L_2 kollarının t=0anındaki başlangıç açılandır, a = [a1 a2 a3 a4 a5]^T ve

b = [b1b2b3b4b5]^T vektörleri ise istenen harekete göre saptanır.

t=0 anına hareketin başladığı an, hareketin sona erdiği 't' anına ise ts denirse açıların başlangıç ve son değerleri $\theta_1(0)$, $\theta_2(0)$, $\theta_1(ts)$, $\theta_2(ts)$ ifadeleri ile gösterilir. Bu açı değerleri trigonometrik olarak bulunabilir.

$\theta_1(0)$, $\theta_1'(ts)$ ve ts değerleri bilindiğinde (verildiğinde) matris eşitliği kullanılarak a = [aı a2 a3 a4 a5]^T vektörü, $\theta_1(0)$, $\theta_1(ts)$ ve ts değerleri bilindiğinde (verildiğinde) ise matris eşitliği kullanılarak b = [b1 b2 b3 b4 b5]^T vektörü bulunur. Bu sonuçlar kullanılarak 'el'in hareket eğrisi çizilebilir.

Robotun 'el' hareketine ilişkin eşitliklerinin tümüye çözülebilmesi için sınır koşullarına ihtiyaç duyulacaktır. Bu koşullar ise; başlangıç anında hızın sıfır ($\theta_1'(0) = v_1(0) =0$) ve ivmenin sıfır ($\theta_1''(0) = w_1(0) =0$) olması, t= ts anında ise hızın sıfır ($\theta_2''(ts) = v_2(ts)=0$) ve ivmenin sıfır ($\theta_1''(ts) = w_1(ts)=0$) olmasıdır.

Hız ve ivme ifadeleri;

$\theta_1'(t) = 5a_1t^4 + 4a_2t^3 + 3a_3t^2 + 2a_4t + a_5$ (1.motor hız ifadesi)

$\theta_1''(t) = 20a_1t^3 + 12a_2t^2 +6a_3t + 2a_4$ (l.motor ivme ifadesi)

$\theta_2''(t) = 5b_lt^4 +4b_2t^3 + 3b_3t^2 + 2b_4t+b_5$ (2.motor hız ifadesi)

$\theta_2''(t) = 20b_1t^3 +12b_2t^2 +6b_3+2b_4$ (2.motor ivme ifadesi)

olur. $\theta_1'(0) = v_1(0) =0$ ilk koşulu $\theta_1'(t)$ eşitliğine uygulanır ise;

$\theta_1'(0) = a_5 = 0$

(1)

bulunur. $\theta_1''(0) = W_1(0) = 0$ İlk koşulu $\theta_1''(t)$ ifadesine uygulanır ise;

$\theta_1''(t) = 2a_4 = 0 \Rightarrow a_4 = 0$

(2)

elde edilir, $t = t_s$ için $\theta_1(t)$ ifadesi;

$\theta_1(tb) = \theta_1(0) + a_1 t_s^5 + a_2 t^4 + a_3 ts^3 + a_4 ts^2 + a_5 ts$

(3)

olur. $\theta'_1(ts) = V_1(t_s) = 0$ koşulu altında,

$\theta'_1(ts) = 5a_1 t_s^4 + 4a_2 t_s 3 + 3a_3 t_s^2 + 2a_4 t_s = 0$

(4)

$\theta''_1(ts) = w_1(t_s) = 0$ koşulu $\theta''_1(ts)$ eşitliğine uygulanır ise;

$\theta''_1(ts) = 20a_1 t_s^3 + 12a_2 t_s^2 + 6a_3 t_s + 2a_4 = 0$ (5)

elde edilir. Yukarıda elde edilen 5 adet eşitlikten;

$$\begin{bmatrix} t_s^5 & t_s^4 & t_s^3 \\ 5t_s^4 & 4t_s^3 & 3t_s^2 \\ 20t_s^3 & 12t_s^2 & 6t_s \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \theta_1(t_s) - \theta_1(0) \\ 0 \\ 0 \end{bmatrix}$$

matris eşitliği elde edilir. Benzer işlemler $\theta_2(t)$ için yapılırsa;

$$\begin{bmatrix} t_s^5 & t_s^4 & t_s^3 \\ 5t_s^4 & 4t_s^3 & 3t_s^2 \\ 20t_s^3 & 12t_s^2 & 6t_s \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \theta_2(t_s) - \theta_2(0) \\ 0 \\ 0 \end{bmatrix}$$

elde edilir. Yukarıda elde edilen iki adet matris eşitliğinin her biri için (bir açı ve onun İki türevinden elde edilen) 3 adet denklem ve 3 adet bilinmeyen bulunmaktadır. Böylece motor kontrolörlerine gönderilecek açı bilgilerine ilişkin polinomun neden 5.dereceden seçildiği de anlaşılmış olmaktadır. En düşük dereceden üç

terim (t^2,t^1,$t°$) katsayıları, t=0 anında sıfır değerini alırlar. Daha yüksek dereceden üç terimin (t^2,t^1,$t°$) katsayıları ise bilinmemektedir. Bu ise üç bilinmeyen anlamına gelmektedir. Eğer polinomun derecesi artırılır ise yeni katsayıları bulmak için ilave sınır koşularına ihtiyaç duyulur.

10.9. Lineer problem çözümüne bir örnek; Girdi-Çıktı analizleri

Nobel Ödülü alan Wassily Leontief ekonominin girdi çıktı konusu üzerine çalışma yapan ünlü bir araştırmacıdır. Lenotiefe göre üretilen ürünlerin tamamı tüketilmelidir. Endüstriyel çıktı talebi 2 adet kaynaktan gelir. a)Farklı endüstrilerin talebi, b)Endüstrilerden başka talep kaynaklan. Buna örnek olarak enerji sektörü seçilebilir;

Enerji şirketleri enerjiyi;
1-Kendi fabrikalarındaki işlemleri gerçekleştirmek için,
2-Başka endüstrilerin elektrik ihtiyaçlarım karşılamak için,
3-Halkın ihtiyaçlarını karşılamak için

üretirler. Buradaki ilk 2 örnek endüstri içi talebi, sonuncusu ise endüstri dışı talebi işaret eder. Girdi çıktı analizleri bir endüstrinin ne kadar ürettiğini ve buna karşılık talebin üretimin ne kadarını tamamı ile karşıladığını açıklar. Bu durumda 'arz ve talebi dengede tutmak için ne kadar üretim yapılmalıdır?' sorusu önem kazanır. İç endüstri talebi girdi-çıktı matrislerinde gösterilebilir. Aşağıda verilen 3*3 boyutlu A matrisi girdi-çıktı matrisine bir örnek olarak verilebilir. Endüstri çıktılarının dolarla ölçüldüğü kabul edilirse aij matris elemanı genel girdi-çıktı matris elemanı olarak adlandırılabilir.

$$\begin{array}{c} \\ \text{Üretici} \end{array} \begin{array}{c} \text{Kullanıcı} \\ 1 \quad 2 \quad 3 \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array} \begin{bmatrix} 0.3 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.3 \\ 0.2 & 0.1 & 0.4 \end{bmatrix} = A$$

A matrisinin eleman değerlerinin ne anlama geldiğini daha iyi anlamak için aij değerlerinden faydalanılabilir. Örneğin a_{12} =0.3 şöyle yorumlanabilir; endüstri 2'nin 1 dolarlık mal çıktısı sağlayabilmesi, endüstri l'e (1 doların %30'u olan) 30 sent'lik çıktı yapma fırsatı verecektir. Fakat bu ilişki tersine çalışmamaktadır. a_{21} = 0.1 olduğu için endüstri 1'in her bir adet dolarlık çıktısı (mal veya hizmet
anlamında) için endüstri 2, 10 sent'lik çıktı verebilmektedir. Bir anlamda A matrisi endüstri içi katkıları gösteren bir büyüklüktür.

Xj, endüstri j'nin (dolar bazında) çıktı miktarı, dj ise endüstri j çıktısı için endüstri dışı talebi göstersin.

Böylece A matris elemanları ve d_j değerlerini kullanarak herhangi bir endüstri üreticisinin çıktısını
hesaplamak mümkün olabilmektedir. Bir endüstrideki, toplam talep endüstri içi ve endüstri dışı talep toplamından meydana gelmektedir. Aşağıda verilen ve A matrisinden faydalanılan lineer denklemi bunu açıklamaktadır:

endüstri içi talep		endüstri dışı talep
$x_1 = 0.3x_1+0.3x_2+0.2x_3$	+	d_1
$x_2 = 0.1x_1+0.2x_2+0.3x_3$	+	d_2
$x_3= 0.2x_1 + 0.1x_2 +0.4x_3$	+	d_3

Eğer yukarıda verilen lineer denklem sistemi yeniden düzenlenir ise;

$d_1 = 0.7x_1 - 0.3x_2 + 0.2\ x_3$
$d_2= -0.1x_ı + 0.8x_2-0.3x_3$
$d_3= -0.2x_1- 0.1x_2 + 0.6x_3$

elde edilir. Yukarıda verilen lineer denklem sisteminden elde edilecek X_j değerleri endüstrideki çıktı dengelerini gösterecektir, aij ve dk değerleri bilindiği sürece endüstriye ilişkin en ekonomik üretim ve tüketim dengesi hesaplanmış olacaktır. Wassily Leontief, bu dengenin sağlanması durumunda ancak en faydalı üretimin yapılabileceğini belirtmiştir. Yukarıda yazılan lineer denklem takımı matrissel formda tekrar düzenlenir ise;

X*=A*X+D; X-A*X=D; (I-A)*X=D

bulunur. Son ifadeden X çekilirse;

X = (I-A)-1*D

elde edilir. Denge denklemi;

D=[50000;30000;60000]

alınmak şartı ile daha önce verilen A matris değerleri kullanılarak X değerleri;

»D=[50000;30000;60000] A=[0.3 0.3 0.2;0.1 0.2 0.3;0.2 0.1 0.4]; ('enter')

»I=eye(3) ; ('enter')

»X=inv(I-A)*D ('enter')

X =

1.0e+005 *

1.7755

1.2735

1.8041

olarak elde edilir. Bulunan X değerleri aşağıda verilmiştir:

$$X = \begin{bmatrix} 177550 \\ 127350 \\ 180410 \end{bmatrix} \$$$

Yukarıda bulunan sonuca göre örneğin, endüstri 1; 177550 $ değerinde çıktı üretmektedir.
Verilen A katsayı matrisi kullanılarak endüstri içi talep (dolar olarak) hesaplanabilir (yukarıdaki MATLAB satırlarına aşağıdaki satır ilave edilip program çalıştırıldığında;

»
ictalep=A*diag (X) ('enter')

ekran görüntüsü;

```
»
  ictalep =  l.0e+004 *

       5.3265    3.8204    3.60S2
       1.7755    2.5469    5.4122
       3.5510    1.2735    7.2163
```

olarak elde edilir. Sonuçlar aşağıda matrisel formda gösterilmiştir:

10.10. Lineer problem çözümüne bir örnek; İşletme maliyeti

Bir inşaat firması A,B ve C tiplerindeki evlerden sırasıyla 10,12 ve 17 'şer adet yapmak üzere taahhüde girmiştir. Evlerin inşaat maliyetini meydana getiren ana unsurlar: çelik, çimento, kereste ve sıhhi tesisat malzemeleri ile işçilik ücretleridir. Her tip evin beheri için hangi malzemelerden ne kadar sarf edileceği ve gerekli işçilik miktarı bilinmektedir. Verilen değerler Tablo 'da gösterilmiştir:

Tablo 10.3

	Maliyet unsurları				
Ev tipi	Çelik	Çimento	Sıhhi tesisat	İşçilik	Kereste
A	10	25	12	22	21
B	12	23	14	26	17
C	11	30	10	18	13

Maliyet unsurlarının birim maliyetleri; çelik için 20 TL, çimento için 13 TL, kereste için 10 TL, sıhhi tesisat malzemeleri için 6 TL ve işçilik için 10 TL olarak verildiğine göre;

a) işin tamamı için her malzemeden ne kadar kullanılacağını,
b) malzemelerin maliyetlerini ye toplam maliyetini bulunuz.
c) aynı işlemleri MATLAB yardımı ile yapınız

Çözüm

a) Yapılacak olan evlerin adetleri tiplerine göre sırası ile P=[10 12 17] vektörü ile ve kullanılan malzemeler de;

$$Q=\begin{bmatrix}10 & 25 & 21 & 12 & 22\\ 12 & 23 & 17 & 14 & 26\\ 11 & 30 & 13 & 10 & 18\end{bmatrix}$$

matrisi ile gösterilebilir. P vektörü ile Q matrisi (P*Q) şeklinde çarpılırsa;

$$P*Q=\begin{bmatrix}10 & 12 & 17\end{bmatrix}\begin{bmatrix}10 & 25 & 21 & 12 & 22\\ 12 & 23 & 17 & 14 & 26\\ 11 & 30 & 13 & 10 & 18\end{bmatrix}=\begin{bmatrix}431 & 1036 & 635 & 458 & 838\end{bmatrix}$$

bulunur. Çarpım vektörü elemanlarının her biri bir malzemenin bütün inşaat için sarf edilmesi gereken miktarlarını verir.
b) Malzemelerin birim maliyetleri sırası ile;
$R = [20\ 13\ 10\ 6\ 15]^T$

sütun vektörü ile gösterilirse, (Q*R) çarpımı her tip evin maliyetim verecektir:

$$Q*R=\begin{bmatrix}10 & 25 & 21 & 12 & 22\\ 12 & 23 & 17 & 14 & 26\\ 11 & 30 & 13 & 10 & 18\end{bmatrix}\begin{bmatrix}20\\ 13\\ 10\\ 6\\ 15\end{bmatrix}=\begin{bmatrix}1137\\ 1183\\ 1070\end{bmatrix}\ TL$$

bulunur. Yani her A tipi ev 1137 TL'ye, B tipi ev; 1183 TL'ye ve C tipi ev; 1070 TL'ye mal olmaktadır. Tüm inşaatın toplam maliyeti P*Q*R çarpımı ile bulunabilir. Bu çarpma (P*Q)*R veya P*(Q*R) şeklinde yapılabilir. Daha kolay olduğu için ikinci yol tercih edilirse, toplam maliyet için,

$$P*(Q*R)=\begin{bmatrix}10 & 12 & 17\end{bmatrix}\begin{bmatrix}1137\\1183\\1070\end{bmatrix}=43756\ \ TL$$

c) Aynı problem MATLAB ile yapılırsa aşağıdaki değerler elde edilir:

```
>>
P*Q
ans =
        4 3 1     10 3 6     6 3 5       4 5 8       8 3 8
Q*R
ans=
        1137
        1183
        1070
P*Q*R
ans=
        43756
```

BÖLÜM 12
EĞRİ UYDURMA, ARA DEĞER VE DIŞ DEĞER HESABI

Ölçüm sonunda elde edilen x değerlerini en yakından temsil eden bir y=f(x) eğrisini bulma işlemi 'eğri uydurma' (curve fitting) olarak adlandırılır. Ölçülen iki x değeri arasında seçilen x_i değerine karşı gelen değerini tahmin etmek ise ' aradeğer hesabı ' (interpolating) olarak adlandırılır. Her iki yaklaşım da deney sonunda elde edilen verilerin değerlendirilmesinde kullanılır fakat yapılan işlem açısından iki yaklaşım arasında farklılık bulunmaktadır. **Ara değer bulma yaklaşımında** kullanılan eğriler, ölçüm sonunda elde edilen **tüm noktadan geçecek şekilde çizilir**. Eğri uydurma yaklaşımında bu şart değildir. Eğer verilen x değer aralığına bakılarak bu aralık dışında bir x_i değerine karşı gelen y_i değeri aranıyor ise bu işlem (extrapolation) dış değer hesabı olarak adlandırılır.

12.1. En küçük kareler metodu ile eğri uydurma

En küçük kareler metodunda çizilen eğri ile ölçülen değerler arasındaki fark en aza indirilmeye çalışılır. Ölçüm sonuçlarından istifade edilerek hesaplanan eğri denkleminin, bir takım ölçüm sonuçlarım (veya tüm ölçüm sonuçlarını) sağlama yükümlülüğü yoktur. Burada amaç; ölçüm sonuçlarına mesafe olarak en az hatalı eğriyi elde etmektir. Eğri uydurma işleminde iki farklı eğri üzerinde çalışılır. Uydurulacak eğrinin denklemi (y=f(x)) doğrusal (lineer) olabileceği gibi (doğrusal olmayan) polinom (veya başka) türde de olabilir.

12.1.1. Doğrusal eğri uydurma

Doğrusal eğri uydurmada kullanılan ortalama karesel hata (OKH) ve bu değerin karekökü;

$$OKH = \frac{1}{N}\sum_{k=1}^{N}(y - y_k)^2 \qquad (12.1)$$

$$\text{KOKH} = \sqrt{OKH} \qquad (12.2)$$

olarak hesaplanır. Yukarıda verilen ifadelerde N; (x_k, y_k) ile gösterilen ölçüm değerlerinin sayısıdır. OKH değeri ise ölçülen y değerleri ile tahmini doğru değeri arasındaki farkın karesinin ortalamasıdır, KOKH ise OKH değerinin kareköküdür.

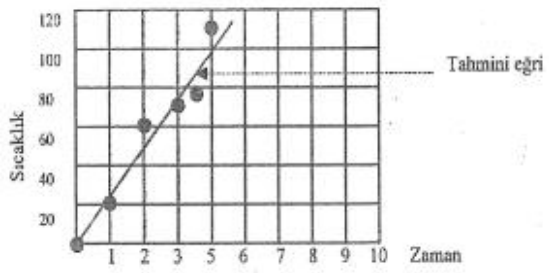
Yapılan bir deney sırasında ölçülen 6 adet sıcaklık değeri ve bu ölçümlerin yapıldığı t zamanları Tablo 12.1 'de verilmiştir.

Tablo 12.1

Zaman (s)	0	1	2	3	4	5
Sıcaklık (F)	0	20	60	68	77	110

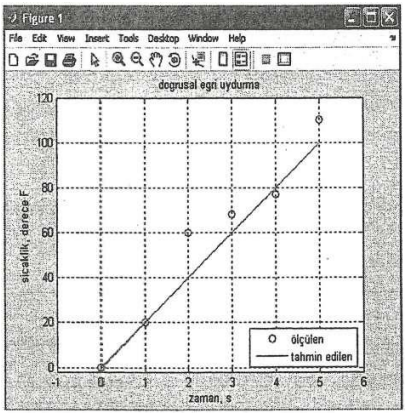
Şekil 12.1'de verilen (siyah) noktalar Tablo 12.1'de verilen ölçüm sonuçlarına dayanarak konulmuştur. Şekil 12.1 'de görülen eğri ise siyah noktalara en yakın geçecek şekilde çizilmeye çalışılan tahmini bir eğridir.

Şekil 12.1'de çizilen tahmini eğriye bakıldığında y=20x denkleminin Tablo 12.1'de verilen x,y değerlerini en iyi temsil edecek (lineer-birinci dereceden) eğri olduğu görülmektedir. Bu eğriyi MATLAB editör ortamında çizdirecek program aşağıda gibi elde edilmiştir (OKH-Ortalama Karesel Hata):



Şekil 12.1

```
x =0:5;
y = [0 20 60 68 77 110]
yciz m 20*x    % tahmini eğri denklemi
hata=yciz-y
OKH=mean (hata.^2)   % % (12.1) eşitliği hesaplanıyor
KOKH=sqrt(OKH) % ortalama karesel hatanın karekoku-(12.2) eşitliği
plot(x,y, 'o' ,x,yciz), title('doğrusal eğri uydurma'),
xlabel(* zaman, s')
ylabel('sicaklik, derece F*) grid
axis([-1,6,-2,120]},
legend(*ölçülen','tahmin edilen',4)
```



Şekil 12.2

MATLAB programından elde edilen çizim şekil 12.2'de verilmiştir. Yukarıdaki programın çalıştırılması sonucunda MATLAB Command Window ortamında elde edilen çıkış satırları aşağıda verilmiştir:

```
»
yciz =
     0    20    40    60    80
hata =
     0     0   -20    -8     3
OKH =
    95.5000
KOKH =
    9.7724
```

Yukarıda yapılan ve tahmini eğri hesaplamalarına dayanan işlemlerin yerine daha doğru ve matematiksel temellere dayanan bir yaklaşım ile meseleye yaklaşmak daha doğru olur. Bunun için ilk adım olarak katsayıları bilinmeyen tahmini bir doğrusal denklem ($y=a_1x + a_2$) yazmaktır. Bu denklemde kullanılan ve a_2 değerleri en küçük kareler yaklaşımı ile bulunur. Bu yaklaşımda kullanılan OKH ifadesi;

$$OKH = \frac{1}{N}\sum_{k=1}^{N}(y - y_k)^2 = \frac{1}{N}\sum_{k=1}^{N}(a_1x_k + a_2 - y_k)^2 \qquad (12.3)$$

olur. Elde edilecek eğrinin en uygun eğri olabilmesi için OKH ifadesinin kullanılan katsayılara (a_1,a_2) göre türevleri sıfır olmalıdır:

$$\frac{\partial OKH}{\partial a_1} = \frac{1}{N}\sum_{k=1}^{N}2(a_1x_k + a_2 - y_k)x_k = \frac{2}{N}\sum_{k=1}^{N}a_1x^2{}_k + a_2x_k - x_ky_k$$

$$\frac{\partial OKH}{\partial a_1} = \frac{2}{N}\left[\left(\sum_{k=1}^{N}x^2{}_k\right)a_1 + \left(\sum_{k=1}^{N}x_k\right)a_2 - \left(\sum_{k=1}^{N}x_ky_k\right)\right] = 0 \qquad (12.4)$$

$$\frac{\partial OKH}{\partial a_2} = \frac{1}{N}\sum_{k=1}^{N}2(a_1x_k + a_2 - y_k)$$

$$\frac{\partial OKH}{\partial a_2} = \frac{2}{N}\left[\left(\sum_{k=1}^{N}x_k\right)a_1 + Na_2 - \left(\sum_{k=1}^{N}y_k\right)\right] = 0 \qquad (12.5)$$

2/N katsayısı ihmal edilerek (12.4) ve (12.5) denklemleri matris formunda yazılırsa;

$$\begin{bmatrix} \left(\sum_{k=1}^{N} x^2{}_k\right) & \left(\sum_{k=1}^{N} x_k\right) \\ \left(\sum_{k=1}^{N} x_k\right) & N \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \left(\sum_{k=1}^{N} x_k y_k\right) \\ \left(\sum_{k=1}^{N} y_k\right) \end{bmatrix} \qquad (12.6)$$

elde edilir. (12.6) normal eşitliğinin çözümü bir önceki bölümde gösterilmişti (ters matris alma, sağdan bölme). (12.6) eşitliğinin çözümünden elde edilecek katsayılar minimum hata ile bulunan katsayılar olacaktır. Aşağıda verilen **MATLAB** komutları teorik alt yapısı yukarıda verilen matematiksel işlemleri yaparak uydurulacak eğriye ilişkin **en uygun katsayı vektörünü** hesaplar.

12.1.2. Doğrusal eğri uydurmaya ilişkin MATLAB komutu

MATLAB ortamında, doğrusal (1. dereceden polinom tipi için) eğri uydurma işleminde kullanılan komut polyfit (x,y, 1) ifadesidir:

a =polyfit(x,y,1) : x ve y vektörleri ile verilen data değerlerinden geçen doğrusal (birinci dereceden) denklemin katsayılarını a vektörüne atar. x ve y vektörlerinin boyutları aynı olmalıdır.

Yukarıda verilen problemi **MATLAB** editör ortamında polyfit komutu yardımı ile çözen program satırları aşağıda gösterilmiştir:

```
x = 0:5; y = [0     20     60     68    77    110];
a = polyfit(x,y, 1)
yciz=polyval(a,x); % hesaplanan a katsayi vektörü yardimi ile
                              % 1.dereceden polinomunun degeri hesaplaniyor
hata=yeiz-y
OKH=mean(hata.^2) KOKH=sqrt(OKH)
plot(x,y,'o',x,yciz), title('dogrusal egri uydurma')
xlabel{'zaman,s'), ylabel(*sicaklik,derece F')
grid,axis{[-1,6,-2,120]), legend('ölçülen',tahmin edilen',4)
```

Yukarıda verilen **MATLAB** programında;

```
a = polyfit{x,y,1)
yciz=polyval(a,x);
```

iki adet komut satırlarının yerine;

```
a=polyval(polyfit(x,y,1),x)
```

komut satırı da yazılabilirdi. Programın çalıştırılması sonunda elde edilen çizim ekranı şekil 12.2'de gösterilmiştir.Yukarıda verilen MATLAB programının Command Window ortamında elde edilen çıkış değerleri aşağıda ki gibi elde edilmiştir:

»

a =

20.8286 3.7619

hata =

3.7619 4.5905 -14.5810 -1.7524 10.0762 -2.0952

OKH =

59.4698

KOKH =

7.7117

»

Yukarıda elde edilen KOKH =7.7117 değeri daha önce elde edilen tahmini KOKH =9.7724 değerinden daha iyi (daha az hatalı) bir sonuçtur. Yukarıda elde edilen a vektörünün değerlerine bakılarak 'uydurulan eğri denklemi';

$y = 20.8286\,x + 3.7619$

olmaktadır.

12.1.3. Doğrusal olmayan (polinom) eğri uydurmaya ilişkin MATLAB komutu

Bu yaklaşım ile yukarıda verilen ve açıklanan 'doğrusal eğri uydurma' yaklaşımı arasında temelde (en küçük kareler metodunu kullanma açısından) bir fark yoktur. Bu iki yöntem arasındaki tek fark bu yaklaşımda uydurulacak olan eğrinin aşağıda gösterildiği gibi;

$f(x) = a_1x^n + a_2x^{n-1} + a_3x^{n-2} + ... + a_nx + a_{n+1}$

n. dereceden bir polinom olmasıdır. **MATLAB** ortamında **doğrusal** olmayan eğri uydurma işlemi için kullanılan komut polyf it (x,y,n) ifadesidir:

a =polyfit (x,y,n): x ve y vektörleri ile verilen data değerlerinden geçen n. dereceden polinomun katsayılarım a vektörüne atar. x ve y vektörlerinin boyutları aynı olmalıdır, n değeri arttıkça, uydurulacak eğri denkleminin daha doğru olduğu söylenemez. _____

'Uydurulacak polinomun' derecesi arttıkça OKH değeri azalır ve eğrinin üzerinden geçtiği noktaların sayısı da artar. n. dereceden bir polinomun katsayı vektörü olan a vektörünün boyutu (sabit katsayı dolayısı ile) ‘n+r’ olacaktır.

Problem 12.1

Yapılan bir deney sonunda 11 farklı değer ortaya çıkmıştır. Bu değerler x-y koordinat sistemine yerleştirildiğinde;

(x_1,y_1)= (0,-0.447) (x_2,y_2) = (0.1,1.978);(x_3,y_3) = (0.2,3.28); (x_3,y_3) = (0.3,6.16)

(x_5,y_5) = (0.4J.08) ; (x_6,y_6) = (0.5,7.34); (x_7,y_7) = (0.6,7.66); (x_8,y_8) = (0.7,9.56)

(x_9,y_9)=(M0.8,9.48); (x_{10},y_{10}) = (0.9,9.3); (x_{11},y_{11}) = (1,11.2)

noktalan elde edilmektedir. Yukarıda verilen 11 adet noktadan geçmeye çalışan 2. mertebeden ve 10. mertebeden iki eğrinin (polinomun) katsayılarını bulan ve her iki eğriyi aynı ekran üzerine çizdiren bir MATLAB programı yazınız.

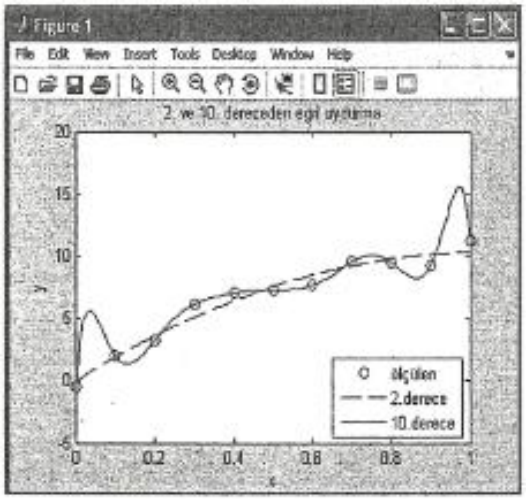
Çözüm

```
x =0:0.1:1;
y =[-0.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];
a2 = polyfit(x,y,2) % polinomun katsayıları bulundu( 2. dereceden)
disp('a2 katsayilari:'}
disp(a2')
a10=polyfit (x,y, 10); % polinomun katsayıları bulundu(10.dereceden)

disp('al0 katsayilari:')
format short e
disp(al0')
xi = linspace{0,1,101);          % x aralığı
yi2 = polyval(a2,xi);            % 2. dereceden polinomun aldığı değerler
                                 % hesaplanıyor
yil0 = polyval(al0,xi);          % 10. dereceden polinomun aldığı değerler hesaplanıyor
plot(x,y,'o',xi,yi2,'—' xi,yil0);% her iki eğri çizdiriliyor
xlabel{'x'), ylabel('y'},title{'2. ve 10. dereceden eğri uydurma');
legend('ölçülen2.derece10.derece',4)
```

Yukandaki **MATLAB** programının sonuçlan aşağıda, elde edilen grafik ise şekil 12.3'de verilmiştir:

```
»
    a2 katsayilari:
   -9.8108e+000
    2.0129e+001
   -3.1671e-Ö02
    al0 katsayilari:
```

-4.6436e+005
2.2965e+006
-4.8773e+006
5.8233e+006
-4.2948e+006
2.0211e+006
-6.0322e+005
1.0896e+005
-l.0626e+004
4.3599e+002
-4.4700e-001



Şekil 12.3

Yukarıda verilen katsayılara bakılarak elde edilen 2 adet polinom denklemi aşağıda verilmiştir:

$y2(x) = -9.8108x^2 + 20.1293x-0.0317$

$yl0(x) =-4.6436*10^5\ x^{10} + 2.2965*10^6x^9 -4.8773*10^6x^8 + 5.823*10^6x^7 -4.294*10^6x^6$

$+ 2.021 * 10^6\ x^5 - 6.0322 *10^5\ x^4 +1.0896 * 10^5\ x^3 -1.0626 * 10^4 * x^2 + 4.3599 * 10^2x - 0.447$

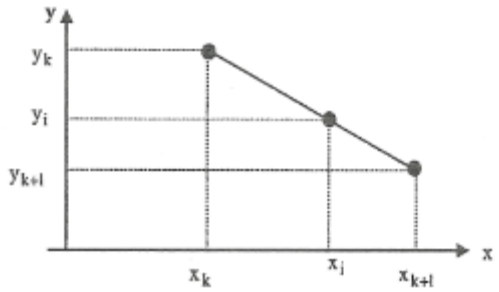
12.2. Ara değer hesabı (interpoiation)

Bir fonksiyonun tablo halinde verilmiş ölçüm sonuçlarından hareketle, bu fonksiyonun bu aralıkta bilinmeyen değerlerinin hesaplanması işlemi 'ara değer hesabı' olarak adlandırılır. Eğer ölçüm sonuçlarının x-y koordinat düzlemindeki dizilişleri bir doğru boyunca ise, aranan ara değerler için 'doğrusal ara değer hesabı' kullanılır. Eğer ölçüm sonuçlarıma x-y koordinat düzlemindeki dizilişleri bir doğru boyunca değil ise 'eğrisel ara değer hesabı* kullanılır. Bu bölümde eğrisel ara değer hesabı için 'kübik (üçüncü dereceden polinom) ara değer hesabı' yaklaşımı kullanılacaktır.

12.2.1. Bir boyutlu doğrusal ara değer hesabı

Şekil 12.5'te birbirini takip eden iki ölçüm sonucu gösterilmiştir, k. ölçüm değeri olan x_k değer için fonksiyonun aldığı değer y_k , (k+1). ölçüm değeri olan x_{k+l} değeri için fonksiyonun aldığı değer ise y_{k+1} olsun. Deneyde i. ölçüm yapılmamış olduğundan x_i ve buna karşı gelen yj değeri hakkında bir fikir istendiğinde 'doğrusal ara değer hesabı' yaklaşımı kullanılarak y; (x_i) değeri hesaplanmaktadır. Şekil 12.4'te verilen $y_k(x_k)$ ve $y_j(x_i)$ değerlerini kullanarak;

$y_i = y_k + m(x_i - x_k)$



Şekil 12.4

yazılabilir. (12.7) eşitliğinde kullanılan 'm' değeri y(x) doğrusunun eğimi olup;

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \qquad (12.8)$$

eşitliği ile hesaplanır. (12.8) ifadesi (12.7)'de yerine konulur ise i.ölçüm değeri;

$$y_i = y_k + \frac{y_{k+1} - y_k}{x_{k+1} - x_k}(x_i - x_k) \qquad (12.9)$$

olarak hesaplanır. Yukarıda da görüldüğü gibi aranan bir değeri bulabilmek için iki adet ölçüm sonucuna ihtiyaç duyulmaktadır. Yukarıda yapılan 'doğrusal ara değer hesabı' işlemi MATLAB ortamında interpl komutu ile gerçekleştirilir:

yi = interpl (x, y, xi) : x ve y vektörleri bilinen ölçüm sonuçlarını içeren eşit boyutta iki vektördür. xi aranan vektör değerlerine karşı düşen yi vektör değerleri, bu komut ile hesaplanır. x vektörünü oluşturan değerler artarak gitmelidir. Aranan xi değerleri x vektörünün sınırları içinde kalmalıdır.
yi=interpl (x,y,xi, 'method'): Doğrusal ara değer hesabında method yerine linear yazılırsa interpl (x, y, xi) komutu ile aynı işlevi görür.

Yapılan bir deneyde x (zaman) değerleri 0 ile 5 arasında birer artarak değişirken bu değerlere karşı gelen y (sıcaklık) değerleri ise y = [0 20 60 68 77 110] vektörü ile değişmektedir. xi =2.6 ve x_t =4.9

değerlerine karşı gelen yi ve y_t değerlerinin bulunması istenmektedir. Aşağıda verilen MATLAB programında bu sorunun cevabı verilmektedir;

```
» x = 0:5;
» y = [0 20 60 68 77 110];
»sicaklik = interpl (x,y, [2.6 4.9])
sicaklik=
        64.8000 106.7000
```

Eğer interpl komutunda y değeri bir vektör olmayıp bir matris olur ise, y değerleri sütun-sütun yerleştirilmelidir. Böyle bir duruma örnek olması için Tablo 10.3'de aynı zaman aralıklarında kaydedilen üç farklı odadaki sıcaklık değerleri verilmiştir. 2.6 saniyedeki her üç farklı odada sıcaklık değerlerinin ne olduğu sorulduğunda bu hesaplamayı yapacak MATLAB programı aşağıda verilmiştir:

Tablo 12.2

Zaman	Sıcaklık 1	Sıcaklık 2	Sıcaklık 3
0	0	0	0
1	20	25	52
2	60	62	90
3	68	67	91
4	77	82	93
5	110	103	96

Not: Tablo 12.2'de 'Zaman' ve 'Sıcaklık' satır sayılarının aynı olduğu görülmelidir, 'y' bir matris ise bu matrisin satır sayısı, x'in eleman sayısına eşit olmalıdır.

```
» x = (0:5) ';                          ('enter')
» y(:,l) = [0,20,60,68,77,110]';        ('enter')
» y(:,2) = [0,25,62,67,82,103]';        ('enter')
» Y(:,3) = [0,52,90,91,93,96]';         ('enter')

» sicaklik = interpl(x,y,2.6)                    ('enter')
  sicaklik=
          64.8000    65.0000    90.6000
```

12.2.2. Doğrusal oimayaıı ara değer hesabı - Kübik yaklaşım

Ara değer hesabında bilinen iki nokta arasındaki değeri hesaplarken doğrusal bir yaklaşım yerine polinom ile ifade edilebilecek bir eğriden de faydalanılabilir. Bu başlık altında incelenecek yaklaşımda polinom olarak üçüncü dereceden (cubic) bir polinom kullanılacaktır.

Yapılan çalışmalara bakıldığında kübik yaklaşım içine spline yaklaşımı da eklenmektedir. Spline yaklaşımı ise şöyle özetlenebilir. Ardarda gelen iki deney sonucu arasında birinci, ikinci yada üçüncü dereceden fonksiyonlarla spline adı verilen yöntem önerilmektedir. Bu yöntem, ölçüm noktalarını çeşitli

aralıklarla bölerek, her bir aralıkta daha küçük dereceden polinomlarla yaklaşım yapma esasına dayanır, spline işleminin mutlaka cubic olarak yapılması gerekmez. Örnek olarak her aralık için seçilen (kübik) fonksiyon;

$$Yi = a_1(x_i - x_k)^3 + a_2(x_i - x_k)^2 + a_3(x_i - x_k) + a_4 \qquad (12.10)$$

olsun. (10.10)'da verilen eşitlikte $x_k < x_i < x_{k+1}$ şartı sağlanmaktadır. Bu eşitlikte kullanılan a_1, a_2, a_3, a_4 katsayıları aşağıdaki şartlardan elde edilir:

- (12.10) eşitliği ölçüm değerlerine ilişkin uç noktasını sağlamalıdır. Örneğin ölçüm sonuçlarından ilki olan x_k değerine karşı gelen y_k ikilisi (12.10) eşitliğini sağlamalıdır. Buna göre $a_4 = y_k$ olmalıdır.
- Bitişik data aralıklarında kübik polinomun (ilktürevi) eğimleri ortak noktalarındaki değerlerine eşit olmalıdır.
- Bitişik polinomlarm eğrilikleri ortak noktalarda aynı olmalıdır.

Kübik-spline ara değer bulma yöntemi MATLAB ortamında spline komutu kullanılarak yapılır;

yi= spline (x,y,xi) : x ve y vektörleri bilinen ölçüm sonuçlarını içeren eşit boyutta iki vektördür. xi aranan vektör değerlerine karşı düşen yi vektör değerleri kübik-spline ara değer bulma yaklaşımı ile hesaplanır. x vektörünü oluşturan değerler artarak gitmelidir. Aranan xi değerleri x vektörünün sınırları içinde kalmalıdır.
yi=interpl (x,y,xi, *nıethod') 'Doğrusal olmayan ara» değer hesabında' method yerine spline veya cubic yazılırsa, spline (x,y,xi) komutu ile aynı işlevi görür.

Yukarıda verilen ölçüm sonuçları cubic-spline yöntemi ile hesaplanır ise;

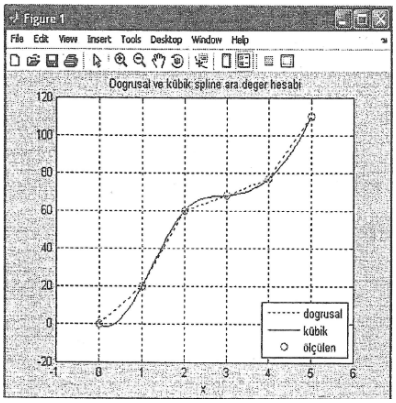
```
» x = 0:5;
» y = [0 20 60 68 77 110];
»sicaklik = spline(x,y,[2.6, 4.9])
  sicaklik =
            67.3013    105.2020
```

elde edilir. Yukarıda verilen ölçüm sonuçlarına dayanarak x=2.6 ve x=4.9 değerlerine karşı gelen y değerleri, hem doğrusal hem de kübik-spline yöntemi ile hesaplanmış oldu. Aynı problemin her iki sonucunu karşılaştırıp çizen MATLAB programı aşağıda verilmiştir:

```
% doğrusal ve kübik-spline yöntemlerinin karsilastirilmasi
x = 0:5;
y = [0 20 60 68 77 110]ı
xi = 0:0.1:5;
ydogru = interpl (x,y,xi) ;
```

```
ykubik = spline(x,y,xi);
plot (xi,ydogru, ':',xi,ykubik,x,y, 'o') ;
legend {x doğrusal', ' kübik', ' ölçülen', 4} ;
title('Doğrusal ve kübik spline ara değer hesabi');
xlabel('x') ;
axis([-1,6,-20,120]); % axis([xmin xmax ymin ymax]) grid;
```

Yukarıda verilen programın çalıştırılması sonunda elde edilen grafik şekil 12.5'de gösterilmiştir.



Şekil 12.5

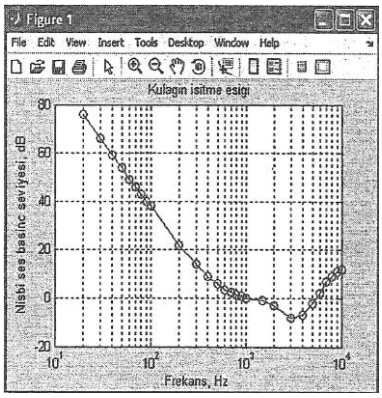
12.2.3. Bir boyutlu ara değer hesabına bir örnek - insanın işitmesi

İnsan kulağının işitme eşiği (ses seviyesinin algılanabilir en alt değeri) frekans ile değişir. Ses basıncının (dB cinsinden) frekans (Hz) ile logaritmik eksende değişimi aşağıda verilmiştir. Bu değerler 0 dB basınç 1000 Hz'de olacak şekilde normalize edilmiştir. Aşağıda verilen MATLAB programı insan kulağının işitme eşiğinin frekans ile değişimini çizmektedir. Elde edilen eğri şekil 12.6'da gösterilmiştir:

```
f = [20:10:100 200:100:1000 1500 2000:1000:10000];   % frekans Hz
spl = [76 66 59 54 49 46 43 40 38 22 …
        14 9 6 3.5 2.5 1.4 0.7 0 -1 -3 …
        - 8 - 7 - 2 2 7 9  11 12]; % ses basinc seviyesi (dB)
        Semilogx(f,spl,'-o');

xlabel('Frekans , Hz');
ylabel('Nisbi ses basinc seviyesi, dB');
title ( Kulagin isitme esigi'),grid on;
```

Şekil 12.6'ya bakıldığında insan kulağının en duyarlı olduğu ses tonunun 3 kHz civarında olduğu görülüyor (sesin duyulması için en az basınç gerekmesinden anlaşılıyor). Aşağıda verilen MATLAB programı 2500 Hz için kulak basıncını üç farklı yaklaşımla hesaplamaktadır. Yukarıda verilen data

değerlerinde 2500 Hz frekansı için 'nisbi basınç seviyesi', ara değer hesabı yaklaşımı ile ve farklı 3 'metot' kullanılarak hesaplansın, 'metot' olarak ise linear, spline, nearest komutları seçilsin:



Şekil 12.6

```
»f=[20:10:100 200:100:1000 1500 2000:1000:10000]; %frekans(Hz)
% spl;ses basinc seviyesi (dB)
»spl = [76 66 59 54 49 46 43 40 38 22 14 9 6 3.5...
        2.5 1.4 0.7  0   - 1   - 3   - 8 - 7 - 2 2 7 9  11 12];
»slineer=interpl(f,spl,2500,'linear') % doğrusal ara değer hesabi
slineer =
        -5.5000
 % cubic spline ara değer hesabi
»skubik=interpl (f, spl,2500, 'spline')
skubik =
        -5.8690
% en yakin komşu ara değer hesabi
»syakin = interpl(f,spl,2500,'nearest'}
syakin =
        -8
```

Yukanda elde edilen üç değer arasında fark bulunmaktadır. En kötü sonuç (doğru değer bilindiği için) **nearest** yaklaşımmdadır. Ara değer hesabmda hangi metodun tercih edileceği önemli bir sorudur. Çoğu uygulamada doğrusal yaklaşım yeterlidir, nearest yaklaşımı kötü yakınsamakla birlikte hızın önemli olduğu (data bilgileri çok sayıda olduğunda MATLAB'm yakınsama hızı azalacaktır) yerlerde tercih edilir.

Zamanı en iyi kullanan yöntem ise spline yaklaşımıdır. Fakat bu yaklaşım bazen istenmeyen sonuçlar üretir.

interpl komutunda 'method' yeri boş bırakıldığında MATLAB'm 'doğrusal ara değer' yaklaşımını (default olarak) kullandığı bilinmelidir.

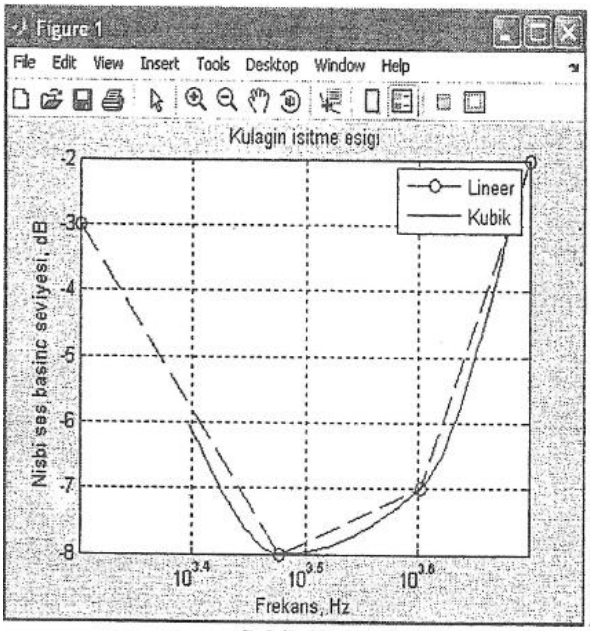
Aşağıda verilen MATLAB programında 'kübik' ve 'doğrusal' yaklaşım kullanılarak basıncın ve frekansın minimum olduğu noktalar tespit edilmektedir:

```
f = [20s 10î100 200:100:1000 1500 2000:1000:100001 ;% frekans (Hz)
                                        % ses basinc seviyesi {dB)
spl = [76 66 59 54 49 46 43 40 38 22 14 9 6 3.5...
        2.5 1.4 0.7 0   - 1   - 3   - 8 - 7 - 2 2 7 9  11 12];
fi = linspace(2500,5000);
spli = interpl(f,spl,fi,'cubic'};          % minimum ara değer hesabi
k = find(f>=2000 & f<=5000);          % minimumdaki indisi bul
                                        % doğrusal ve kübik data yi ciz
semilogx(f (k) ,spl(k),'--o' , fi, spli, 'k-'M , legend( 'Lineer', 'Kübik')
xlabel('Frekans, Hz')
ylabel('Nisbi ses basinc seviyesi, dB')
title('Kulagin işitme esigi')
grid on
[splmin,kmin] = min(spli); % minimum değer ve bunun indisi
splmin
kmin
fmin = fi(kmin);          %minimum indise karsi gelen frekans
fmin
```

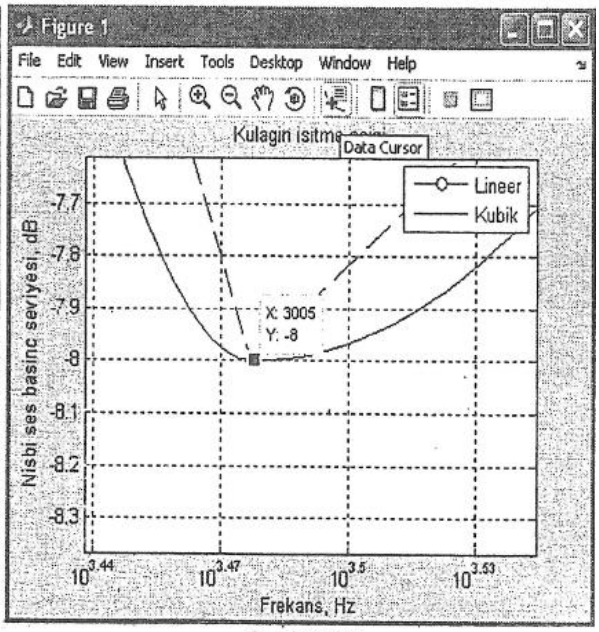
Yukandaki MATLAB programının uygulanması sonunda elde edilen değerler aşağıda verilmiştir:

```
»
splmin =
        -8.0000
kmin =
        21
fmin =
        3.0051e+003
```

Elde edilen sonuçlara bakarak insan kulağının en duyarlı olduğu tonun 3 kHz yakınında olduğu görülmektedir. Yukarıda verilen programın uygulanması ile elde edilen değişim şekil 12.7'de gösterilmiştir. Eğrinin minimum olduğu noktayı bulmak için önce şekil 12.7'de **Zoom In** ikonuna 'tıklanıldığında fare yardımı ile hassas bölge kutu içine alınır. Bu işlem sonunda hassas bölge daha iyi bir görüntüye kavuşur. Daha sonra **Data Cursor** ikonuna 'tıklanılarak aktif hale getirilir. Şekil 12.8'de minimum gibi gözüken noktaya 'tıklanıldığında ise bu noktanın koordinatlarını veren pencere ile karşılaşılır. Bu sonuca bakarak, kübik yaklaşım ile aranan noktaya daha iyi yaklaşıldığı görülmektedir.



Şekil 12.7



Şekil 12.8

12.3. İki boyutlu ara değer hesabı

Bir verinin diğer iki değişkenin fonksiyonu olduğu durumlarda bu iki verinin bazı değerlerine karşılık gelen fonksiyon değerini bulmak için interp2 komutu kullanılır;

interp2 {x,y, z,xi,yi, 'method') : Bu komutun MATLAB ortamında çalışması interpl komutuna benzer şekildedir, interpl komutunda y=f(x)- bir boyutlu fonksiyon- iken, burada z=f(x,y)-iki boyutlu fonksiyon- özelliğine sahiptir, 'method' olarak ise nearest, spline, linear veya cubic yaklaşımları kullanılabilir. Tüm bu metodları uygulayabilmek için gerekli şart, x ve y vektörlerinin ya hep artan ya da hep azalan verilerden oluşması gerektiğidir. x ve y verileri arasındaki aralıkların düzenli bir şekilde artması ya da azalması gerekli değildir. x ve y dizilerinin elemanları arasında eşit aralık bulunuyor ise daha hızlı ara değer hesabı için linear veya cubic yöntemlerden biri seçilmelidir. Burada verilen x, y ve matrislerinden yola çıkılarak z matrisi, bilinen xi ve yi matrislerinden yola çıkılarak ise zi matrisi (ara değer yaklaşımı kullanılarak) bulunur. Eğer x bir vektör ise x'in eleman sayısı z matrisinin sütun sayısına eşit olmalıdır. Benzer şekilde y bir sütun vektör olabilir. Bu durumda da y'nin elemanları z'nin satır sayısına eşit olmalıdır.

Problem 12.2

Okyanus zeminin haritasını çıkartmakla görevli bir araştırma şirketinin okyanus içinde sonar cihazı kullanarak 0.5 km aralıkla yaptığı çalışma sonunda her (x,y) çiftine karşı gelen derinlik miktarları (z ekseni) tespit edilmiştir. Ölçüm sonuçları tespit edilirken her x noktası için y ekseninde 0.5 km aralıkla 13 Ölçüm yapılmaktadır. x ekseni boyunca ise 0.5 km aralıkla 4 km mesafe alınmaktadır. Bu şekilde z ekseni için 117 değer tespit edilmiştir. Okyanusun en derin noktasını interp2 komutu yardımı ile ara değer hesabı yaparak bulacak MATLAB programı yazınız ve okyanus zeminini elde edilen değerler yardımı ile çizdiriniz.

Çözüm x=0:0.5:4; y=0:0.5:6;

```
z =[  100  99  100  99  100  99  99  99  100;
      100  99  99   99  100  99  100 99  99;
      99   99  98   98  100  99  100 100 100;
      100  98  97   97  99   100 100 100 99;
      101  100 98   98  100  102 103 100 100;
      102  103 101  100 102  106 104 101 100;
      99   102 100  100 103  108 106 101 99;
      97   99  100  100 102  105 103 101 100;
      100  102 103  101 102  103 102 100 99;
      100  102 103  102 101  101 100 99  99;
      100  100 101  101 100  100 100 99  99;
      100  100 100  100 100  99  99  99  99;
      100  100 100  99  99   100 99  100 99] ;
mesh(x,y,z);
xlabel('x ekseni- km'), ylabel ('y ekseni- km');
zlabel{'okyanus derinliği- km'); title('okyanus derinliğinin ölçülmesi');
```

Yukarıda verilen MATLAB programının uygulanması ile elde edilen grafik şekil 12.8(a)'de gösterilmiştir. Grafik incelendiğinde en derin yerin x=2.5 km, y=3 km civarlarında olduğu görülmektedir. Kullanıcının bu iki değere ulaşmak için şekil 12.9'da gösterilen şekil penceresinde Zoom In seçeneğini 'tıklayarak maksimum nokta civarını fare ile işaretlemeli ve bölgeyi daha iyi görecek şekilde ayarlamalıdır. Daha sonra ise Data Cusor ikonu yardımı ile maksimum nokta civarına tıklayarak aranılan noktanın koordinatlarını açılan küçük pencereden okumalıdır (şekil 12.19). Kullanıcı fare hareketi ile x eksenini tam karşısına alırsa (y ekseni bu durumda gözükmez) en derin yerin x değerini büyük bir yaklaşıklıkla görebilir. Aynı şey y ekseni içinde yapılırsa (bu durumda x ekseni gözükmez) en derin yerin y değeri büyük bir yaklaşıklıkla görülebilir. Bu iki ayn işlemi yapmak yerine kullanıcı z eksenini döndürerek de şekil penceresinden en derin (z) değeri tespit edebilir.

Aşağıda yazılan MATLAB satırı ile (2.5,3) koordinatları için z değeri hesaplanmaktadır; **(Not:** Programda x ve y vektörlerinin boyutu ile z matrisinin satır ve sütun sayıları arasındaki ilişki mutlaka bilinmelidir, z'in sütun sayısının x'in boyutuna, z'in sütun sayısının ise y'nin boyutuna eşit olduğu fark edilmelidir.)

»zmax=interp2 (x, y, z, 2.5,3) ('enter')

zmax =

108

Kullanıcı, yukarıdaki MATLAB satırında, 2,5 ve 3 değerlerini değiştirerek maksimum derinliği arayabilir.

MATLAB komutları içinde interp2 komutu ile aynı amaca hizmet eden griddata komutu bulunmaktadır:

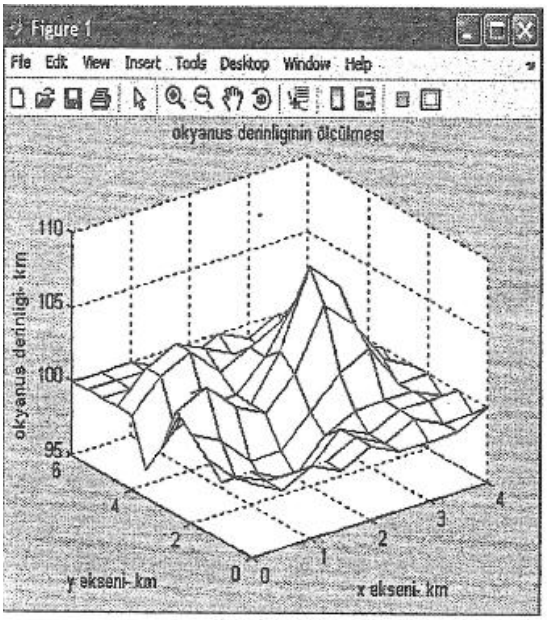
»zl=**griddata** (x,y, z,xl,yl) ('enter')

Yukanda verilen örnekte kullanımı gösterilen griddata komutu (daha önce tanımlanmıştı), x ve y vektörlerini kullanarak (x_l,y_l) çiftine karşı gelen z_l değerini bulur. Bu işlemi yaparken x, y ve z verileri ile uydurulan yüzey denkleminden faydalanır. Bir önceki problem için bu komut kullanılırsa;

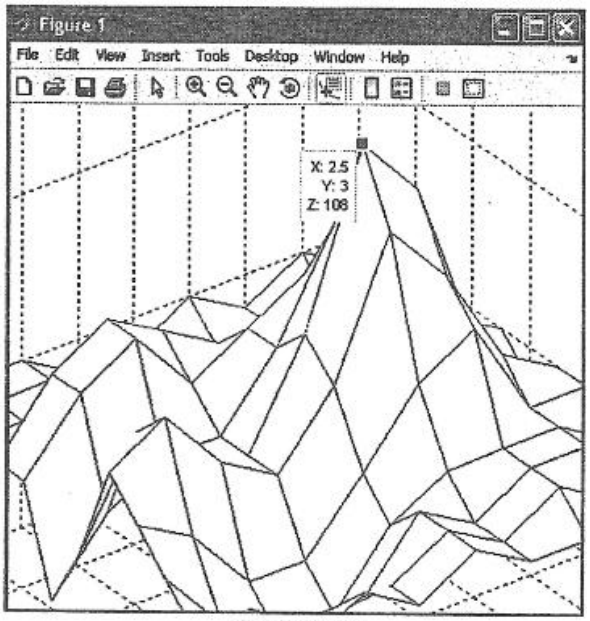
»zmax=griddata (x,y, z, 2.5,3) ('enter')

zmax = 108

bulunur. Görüldüğü gibi iki sonuç birbirlerine oldukça yalandır.



Şekil 12.9



Şekil 12.10

12.4. Üç boyutlu ara değer hesabı

Bir veri grubunun üç farklı değişkenin fonksiyonu ve 3 boyutlu bir dizi oluşturması durumunda üç verinin bazı değerlerine karşı gelen değeri bulmak için interp3 komutu kullanılır. Bu komutun genel kullanımı;

»vl=**ınterp3** (x,y,z,v,xl,yl,zl) ('enter')

şeklindedir. Bu komutun kullanılış biçimi interp2 ile aynıdır ve aynı çözüm yöntemleri kullanılır. x, y ve z aynı eleman sayısına sahip vektörler olmalıdır. Aynı uzunlukta olmayan x, y ve z vektörlerinin bulunması halinde meshgrid komutu kullanılarak gerekli uygunlaştırma işlem yapılabilir.

xii **n boyutlu ara değer hesabı**

Bir veri grubunun n farklı değişkenin fonksiyonu ve n boyutlu bir dizi oluşturması durumunda, n adet verinin bazı değerlerine karşı gelen değeri bulmak için interpn komutu kullanılır, 'linear', 'cubic' 'nearest[1], 'spline' seçenekleri 'method' adı altında bu komut için de geçerlidir. Komutun genel kullanımı aşağıdaki satırda gösterildiği biçimdedir.

»vl=interpn (xl,x2,x3,……,v,yl,y2,y3, ...) ('enter')

xiii**Dış değer hesabı (extrapolasyon)**

interpl (x, y, xi, 'method', [1] extrap[1]) : Bir boyutlu dış değer hesabı yapar.

interp2 (x, y, z, xi, yi, [x] method', [1] extrap') : İki boyutlu dış değer hesabı yapar.

interpn (xl, x2, x3, ..., v, yl, y2, y3, . .., [1] method' , [1] extrap') : n boyutlu dış değer hesabı yapar.

Aşağıda, tablo 12.1'de verilen değerlere İlişkin dış değer hesabı gösterilmiştir. x=6 ve x=9 değerlerinin x= 0 **:** 5 vektörünün dışında kaldığına dikkat edilmelidir:

» x = 0:5;

» y = [0 20 60 68 77 110]; ('enter')

» sicaklik = interpl (x,y, [6 9], ' linearextrap') ('enter')

sicaklik =

143 242 % cikan sonuçlar da y vektörünün disinda kalmaktadir

xiv **Eğri uydurma işleminin Basic Fitting arayüzü yardımı ile gerçekleştirilmesi**

Eğri uydurma işleminin Basic Fitting arayüzü yardımı ile nasıl gerçekleştirildiği aşağıda verilen problemin çözümü üzerinden anlatılacaktır.

Problem 12.3

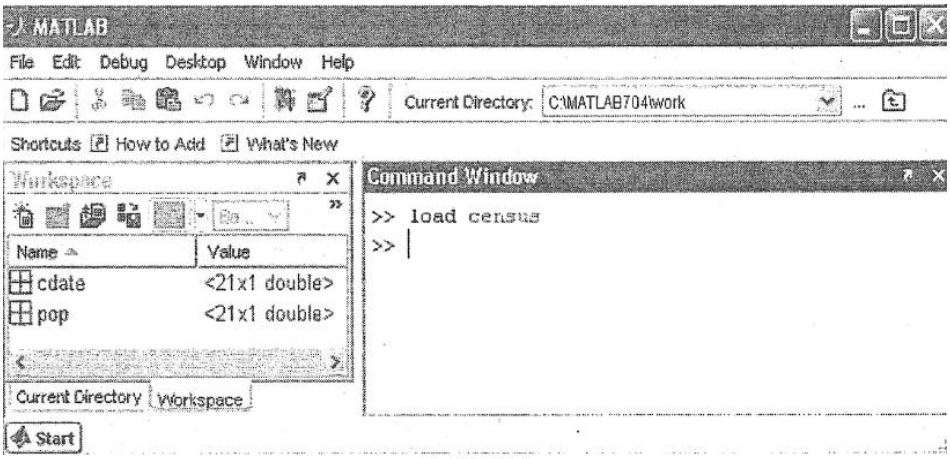
MATLAB ortamında (arka planda) tanımlı olan census adlı data dosyası 1790 ile 1990 yılları arasında ABD'deki yaş populasyonunu içermektedir. Bu değişimi temsil edecek 4. dereceden polinomun katsayılarını bulunuz.

Çözüm

Command Window ortamında;

» load census ('enter')

komutu uygulandığında Workspace ortamında cdate ve pop adlı iki vektör matris ortaya çıkacaktır (şekil 12.11) . cdate; 1790 ile 1990 arasındaki 10'ar yıllık artışları gösteren sütun vektör, pop; 1790 ile 1990 yıllan arasındaki popülasyonu gösteren sütun vektördür.



Sekil 12.11

Şekil 12.12'de

» plot(cdate,pop,'ko') ('enter')

komutunun uygulanması sonunda elde edilen pop=f(cdate) değişimi gösterilmiştir. Aşağıda verilen komut satırı uygulandığında bu iki vektör arasındaki ilişkiyi temsil eden 4. dereceden bir polinomun katsayıları elde edilecektir:

» p=polyf it (cdate, pop, 4) ('enter')

Warning: Polynomial is badly conditioned. Remove repeated data points

or try centering and scaling as described in HELP POLYFIT.

»In polyfit at 81

p =

Columns 1 through 3

4.7543e-008 -3.5557e-004 1.0032e+000

Columns 4 through 5

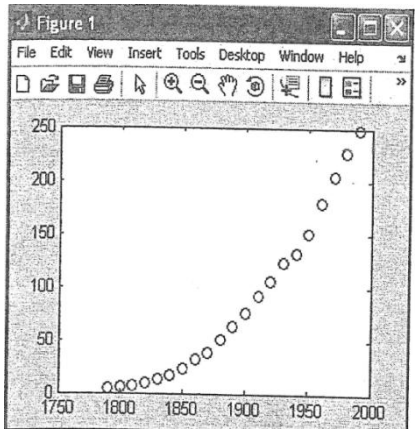
-1.2644e+003 6.0020e+005

Yukarıda verilen uyarıda (her ne kadar polinomun katsayıları elde edilmiş olsa da) cdate ve pop adlı iki vektör arasındaki dağılımının pek uygun olmadığı, verilen data değerlerinin normalize edilerek eğri uydurma işleminin tekrar yapılması önerilmektedir

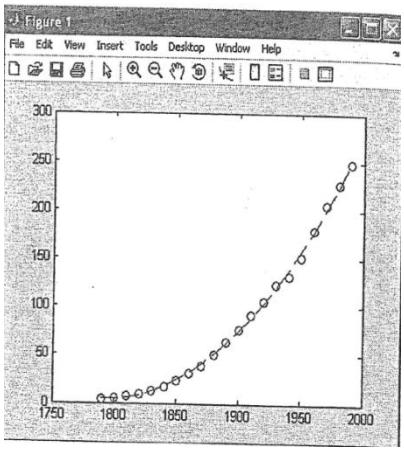
Normalizasyon işlemi için önerilen yollardan bir tanesi de; cdate değerleri ile cdate değerlerinin ortalaması arasındaki farkı, cdate değerlerinin standart sapmasına bölmektir. Aşağıda bu işlemin yapıldığı MATLAB komut satırı gösterilmiştir:

»sdate={cdate-mean(cdate)} ./std(cdate); ('enter')

Yukanda verilen komut satırının uygulanması sonunda elde edilen sdate adlı vektör matrisi, cdate adlı vektör matrisinin açıklanan normalizasyon yaklaşımına maruz bırakıldığında elde edilen (normalleştirilmiş) değerleridir.



Şekil 12.12



Şekil 12.13

cdate adlı vektörün normalleştirilmesi sonunda elde edilen sdate adlı vektöre polyfit komutu uygulandığında;

»p=polyf it (sdate, pop, 4) ('enter')

P =

7.0471e-001 9.2102e-Q01 2.3471e+001 7.3860e+001 6.2229e+001

elde edilir. 4. dereceden bu polinom için sdate vektörünün aldığı değerler;

»pop4=polyval (p, sdate) ;

komut satırı ile bulunabilir. Böylece normalizasyon işlemine maruz bırakılmış cdate değerleri ile normalizasyon işlemi yapılmamış cdate değerleri aym eksen takımı üzerinde aşağıdaki komut satın yardımı ile çizdirilirse şekil 12.13'de görülen değişimler elde edilir:

»plot(cdate,pop4,* cdate,pop, ‘o’)

12.7.1. Eğri uydurmada kullanılan arayüzlerin tanıtılması

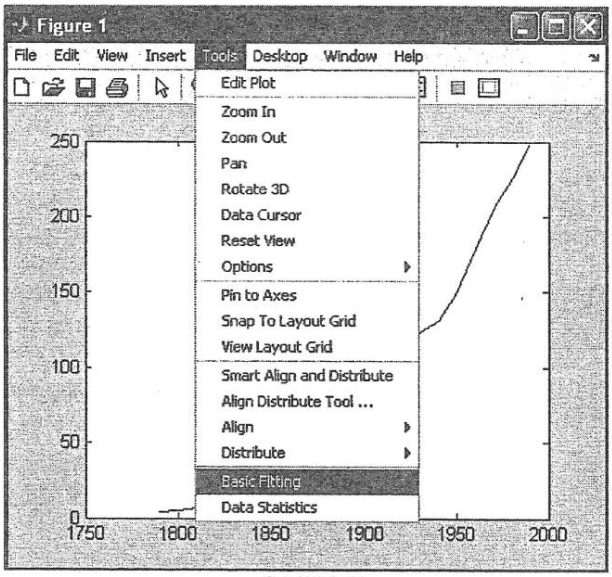
Kullanıcının deney sonucu gözlemlediği data değerleri ile (çeşitli yaklaşımlar kullanarak) eğri uydurma yaklaşımı sonucu hesapladığı değerler arasındaki ilişkinin doğruluğunu tespit etmek

için kullanılan yaklaşım şekillerinden bir tanesi rezidü hesabıdır. MATLAB ortamında normalize edilmiş data değerlerinin çeşitli eğri uydurma yaklaşımlarına uygulanması sonunda elde edilen sonuçlar rezidü hesabı yapılarak karşılaştırılabilir.

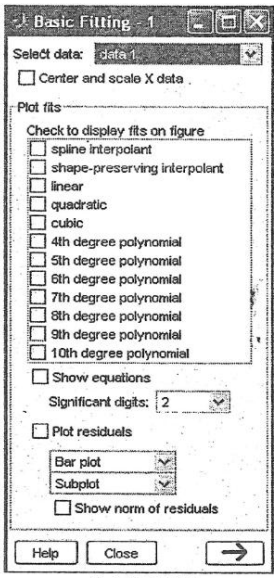
MATLAB ortamında eğri uydurma amaçlı kullanılan iki farklı arayüz ortamı bulunmaktadır. Bunlardan ilki **Basic Fitting** arayüzü diğeri ise **Curve Fitting Tool** arayüzüdür.

12.7.1.1. Eğri uydurma işleminde Basic Fitting arayüzii

Yukarıda verilen cdate ve pop adlı iki vektör arasındaki eğri uydurma işlemi, **MATLAB** ortamındaki **Basic Fitting** arayüzii kullanılarak da çok kolay bir şekilde yapılabilir. Bunun için aşağıdaki işlemlerin sıra ile gerçekleştirilmesi gerekir:



Şekil 12.14



Şekil 12.15

xv Verilen data değerleri plot komutu yardımı ile çizdirilmelidir:

» load census ('enter')

» p=plot(cdate,pop) ('enter')

Elde edilen şekil penceresinde Tools seçeneği içine girilerek Basic Fitting üzerine 'tık'lanılmalıdır (şekil 12.14). Bu işlem yapıldığında şekil 12.15'de gösterilen pencere ile karşılaşılır.

xvi Şekil 12.15'de gösterilen Basic Fitting penceresinde, Select Data ifadesinin yanındaki boşlukta, şekil 12.14'de verilen eğriye ilişkin data değerlerinin datai adlı ortamda saklandığı anlatılmaktadır. Center and scale X data yazısının sol yanında yer alan boş kutu üzerine 'tıklanıldığında datai adlı değişimin normaiizasyon işlemi (arka planda) yapılır.

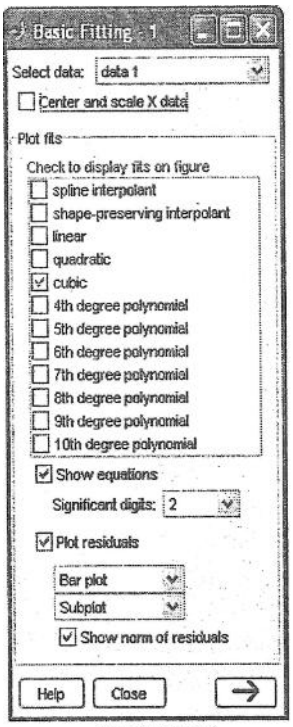
xvii Check to display fits on figure yazısının altında kalan pencere içinde çeşitli seçenekler yer almaktadır. Bu pencere içinde kullanıcıya iki farklı tip eğri uydurma seçeneği sunulmaktadır: Bu iki seçenek; interpolasvon (ara deðer bulma) ve polinom seçimidir. Bu pencere içinde yer alan kutucuklardan ilki spline interpolant seçeneğidir. Bu kutucuk işaretlendiğinde **MATLAB** hesaplamaları spline komutu kullanılarak gerçekleştirilir. Bu kutucuğun altında kalan kutucuk işaretlendiğinde ise kutucuklardan ilki spline interpolant seçeneğidir. Bu kutucuk

işaretlendiğinde MATLAB hesaplamaları pchip komutu kullanılarak gerçekleştirilir. Bu komut MATLAB arka planında 'Piecewise Cubic Hermite Interpolating Polynomial' adlı interpolasyon metodunu kullanır. Bu pencere içinde baştan üçüncü kutu ve altında yer alan diğer kutular işaretlendiğinde ise sırası ile birinci (linear), ikinci (quadratic), üçüncü (cubic), dereceden polinomlar yardımı ile eğri uydurma işlemi gerçekleştirilir. Eğer verilen data değerleri N adet ölçüm içeriyor ise MATLAB ortamında en fazla N. dereceden bîr polinom uydurmaya izin verilir.

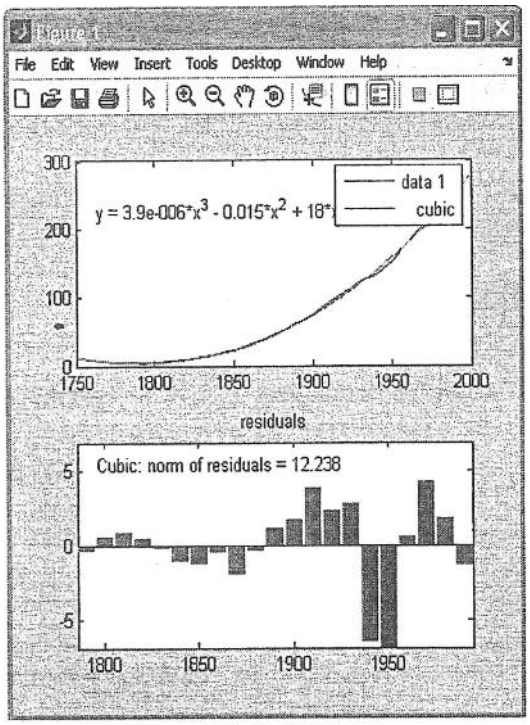
xviii Show equation ifadesinin sol yanında yer alan kutucuk işaretlendiğinde ise eğri uydurma yaklaşımında kullanılan polinomun denklemi ekran üzerinde gözükür. Significant digits seçeneğinin yanında yer alan boşlukta ise bu polinomun katsayılarından kaç adet önemli sayının gösterileceği belirlenir.

xix Plot residuals seçeneği işaretlendiğinde bu ifadenin altında yer alan iki seçenek önem kazanır. İlk seçenek (Bar plot yazan) rezidü değişiminin ne şekilde çizdirileceği (Bar plot, Scatter plot, Line plot) hakkında kullanıcının tercihini sormaktadır. Bunun altında yer alan seçenekte ise (Subplot yazan) çizdirilecek rezidü eğrisinin mevcut olan şekil penceresinin altına mı (Subplot) yoksa bağımsız bir pencere (Separate) olarak mı çizdirileceği konusunda kullanıcıya seçenek verir.

xx Show norm of residuals seçeneği yanında yer alan kutucuk 'tık'lanıldığında ise kullanıcı (hesaplamalarda hata ile orantılı olan) rezidü genliği hakkında kullanıcıya bilgi verilir.



Şekil 12.16



Şekil 12.17

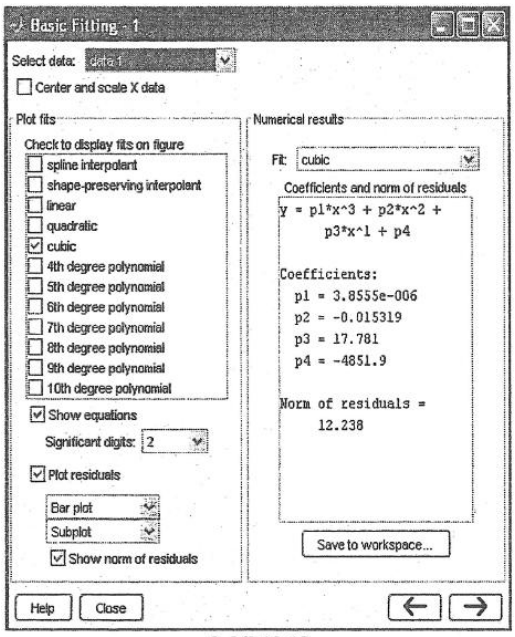
Yukarıda yapılan açıklamalara örnek olması bakımından şekil 12.15'de işaretlenen kutucukları gösteren pencere şekil 12.16'da verilmiştir. Şekil 12.16'da gösterilen seçeneklerin işaretlenmesi sonunda elde edilen ekran görüntüsü ise şekil 12.17'de gösterilmiştir.

```
» plot(cdate,pop,'ro')
```

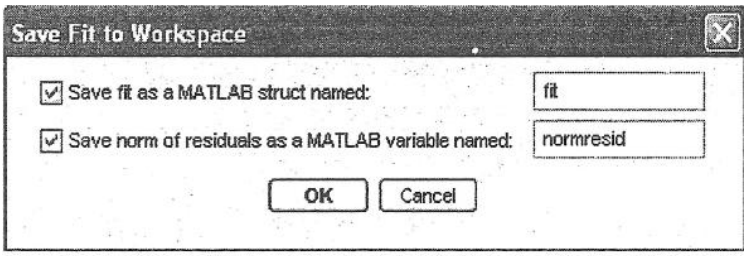
Şekil 12.16'da a butonuna 'tık'lanılır ise Şekil 12.18 'de verilen ekran görüntüsü ile karşılaşılır.

Şekil 12,18'de verilen pencerenin sağ tarafında yer alan Numerical results bölümünde sırası ile; uydurulan polinom denklemi, polinomon (en yüksek dereceden başlayarak sıralanan) katsayıları ve rezidü normu (genliği) verilir. Bu bölümüm en altında yer alan Save to workspace seçeneği 'tık'lanıldığında ise kullanıcının karşısına şekil 12.19'da gösterilen pencere çıkar.

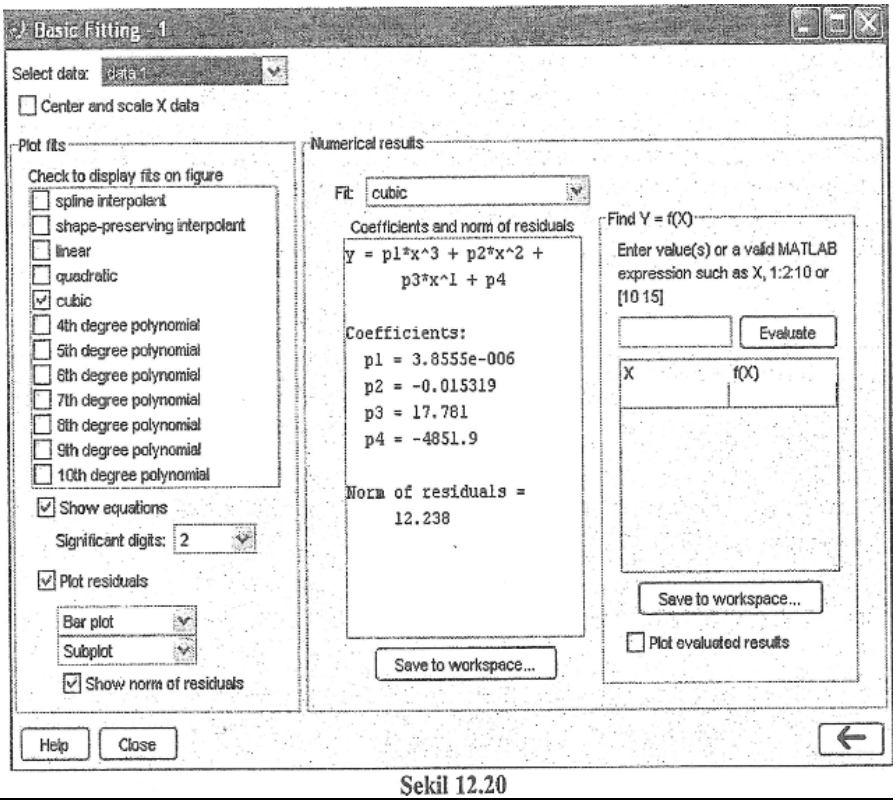
Şekil 12.19'da verilen pencerede (fit ve normresid) adları kullanıcı tarafından istenirse değiştirilebilir) OK seçeneği 'tık'lanıldığında, Workspace ortamında bu adlar ile anılan bir yapı ve bir değişken oluşturulur, `fit` adlı yapı içinde polinomun derecesi ve katsayılarını bildiren bilgiler, normresid adlı değişken içine ise rezidü normu yazılır. Eğer şekil 12.18'de gösterilen pencere içinde sağ en altta yer alan E3 butonuna 'tık'lanılır ise şekil 12.20 ile verilen ekran görüntüsü elde edilir.



Şekil 12.18



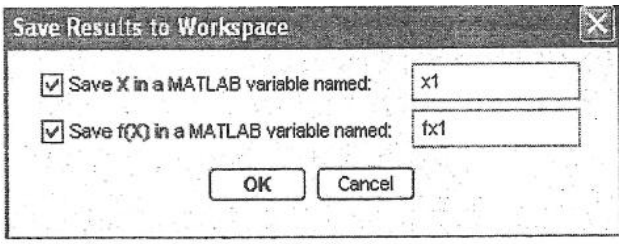
Şekil 12.19



Şekil 12.20

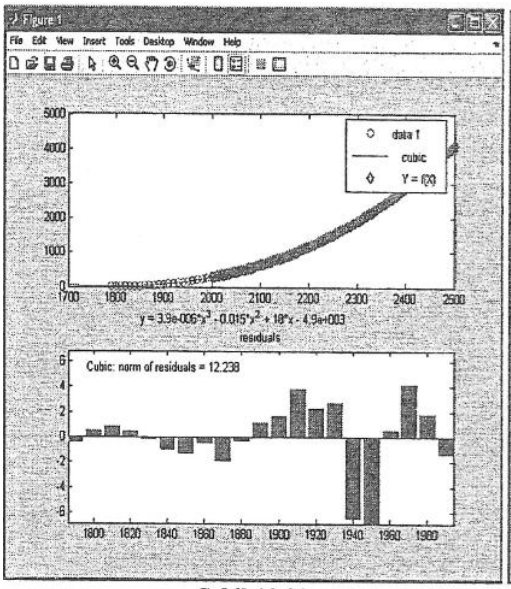
Şekil 12.20'de verilen Find Y=f (X) pencerenin sağ tarafında yer alan Evaluate seçeneğinin sol yanında yer alan boşluğa, ölçüm sonunda hakkında bilgi sahibi olunmayan edat e değerleri girilir. Evaluate seçeneği 'tık'lanılır ise en üstünde X ve f (X) ifadeleri bulunan boş alana cdate değerlerine karşı gelen pop değerleri gelir (hesaplanır). Kullanıcı Evaluate seçeneğinin sol tarafındaki boşluğa (1800:5:1850 biçiminde) vektör biçiminde aradığı cdate değerlerini de girebilir.

Kullanıcı, Evaluate seçeneğinin sol tarafındaki boşluğa ölçümde göz önüne alman en büyük cdate (yani X) değerlerlerinden daha büyük değerler girerek ekstrapolasyoıı da yapabilir. Ekstrapolasyon sonunda elde edilen yeni pop (yani f(X)) değerleri en üstünde X ve f (X) ifadeleri bulunan boş alana yazılır. Eğer sağ en altta yer alan Save to workspace seçeneği işaretlenir ise elde edilen sonuçlar Workspace ortamında kullanıcının belirleyeceği bir isim altında ayrı ayrı kaydedilir. Bu seçenek kullanıldığında kullanıcının karşısına çıkan ekran görüntüsü şekil 12.2l'de verilmiştir. Kullanıcı isterse xl ve fxl adlarını değiştirebilir.

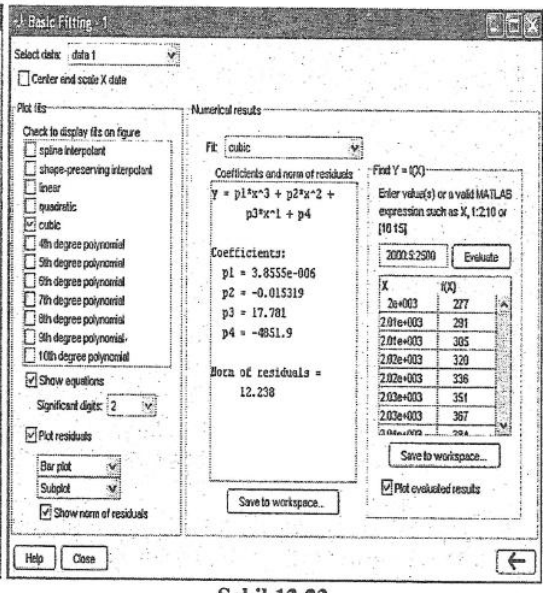


Şekil 12.21

Eğer kullanıcı bu yeni değerleri eğri üzerinde görmek isterse, şekil 12.20'de yer alan Plot evaluate resul t s seçeneğini işaretlemesi yeterlidir. Burada bir ekstrapolasyon örneği olması için Evaluate seçeneğinin sol tarafındaki boşluğa 2000:5:2500 yazılmış, Plot evaluate results seçeneği işaretlenmiş ve elde edilen değişim şekil 12.22'de gösterilmiştir. Şekil 12.20'nin ekstrapolasyon uygulaması sonunda aldığı görüntü ise şekil 12.23'de gösterilmiştir.



Şekil 12.22



Şekil 12.23

Kullanıcı isterse şekil 12.20'de Plot Fi t s penceresi içinde yer alan eğri türlerinden birkaç tanesini aynı anda işaretleyebilir. Bu durumda şekil 12.22'de birden fazla eğri ve rezidü değeri elde edilir. Kullanıcı isterse en uygun olan eğriyi bunların içinden seçebilir.

12.7.1.2. Eğri uydurma işleminin Curve Fitting Tool arayüzü yardımı ile gerçekleştirilmesi

MATLAB araç kutuları içinde eğri uydurma işleminin yapılması için geliştirilen diğer bir arayüz (GUI) Curve Fitting Tool ortamıdır. Bu ortamı kullanarak; bir veya daha fazla data seti incelenebilir. Grafik olarak en iyi eğri uydurma işlemi, aynı datayı birden fazla eğri ile çizdirip karşılaştırma imkanı, ara değer

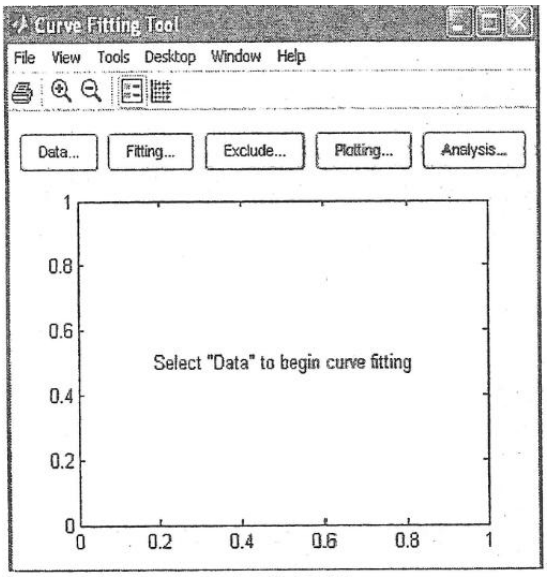
bulma-dış değer bulma-türevsel ve entegral eğri uydurma işlemleri yapılabilir. Eğri uydurma araç kutusunu çalıştırmak için aşağıdaki komut satırının uygulanması yeterlidir:

»cftool ('enter')

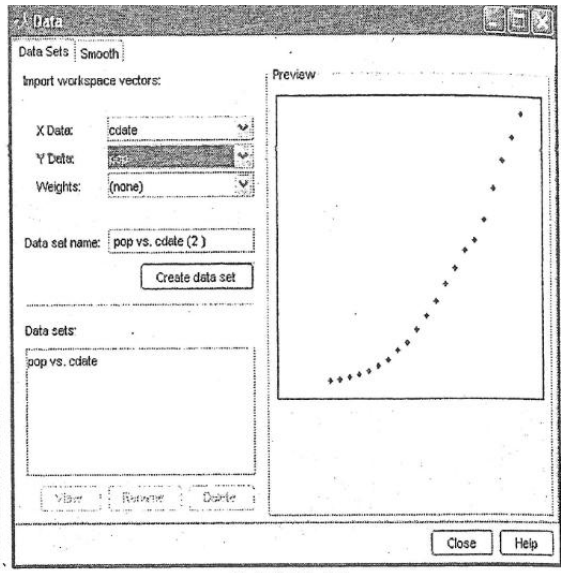
Yukarıdaki komut satırı uygulandığında kullanıcının karşısına şekil 12.24'de verilen görüntü çıkar. Bu araç kutusunun tanıtılması için örnek olarak yine problem 12.3 çözülecektir: Bölüm 12.6.1.1'de belirtildiği gibi;

»load census ('enter')

komutu uygulandığında Workspace ortamında şekil 12.11'de gösterildiği gibi cdate ve pop adlarında iki adet vektör oluşur (cdate; x ekseni ve pop; y ekseni seçilecektir). Önce bu iki vektörün cftool ortamına taşınması açıklanacaktır.



Şekil 12.24



Şekil 12.25

12.7.1.2.1. Workspace ortamındaki datanın eğri uydurma araç kutusuna taşınması (importing)

Şekil 12.24'de Data. .butonuna 'tık'lanıldığında (veya File menüsü altında yer alan Import Data.. komutuna 'tık'lanıldığmda) şekil 12.25'de gösterilen pencere açılır. Bu pencere içinde **x** Date ifadesinin sağ tarafındaki ok işaretine 'tık'lanıldığmda Workspace ortamında saklı olan büyüklükler ile karşılaşılır. Bunlar içinden x eksenine karşı gelmesi düşünülen vektör işaretlenir. Benzer şeyler Y Date ifadesi için geçerlidir. Bu işlemler gerçekleştirildiğinde şekil 12.25'de Preview penceresinde, bu iki vektörün değişimi görülecektir. Bu gösterimde yalnızca x ve y değerlerinin işaretlendiği, her hangi ibr eğri çizim işleminin yapılmadığına dikkat edilmelidir.

Şekil 12.25'de Create data set komutuna 'tık'lanıldığmda cdate ve pop vektörleri Curve Fitting **Tool** araç kutusuna taşınır (importing). Bu işlem sonunda şekil 12.24'de verilen Curve Fitting **Tool** penceresi şekil 12.26'da gösterilen forma dönüşür. Artık eğri uydurma işlemine geçilebilir.

Şekil 12.25'de Weights (ağırlık sabitleri) ifadesinin sağ tarafındaki boşluğa (varsa) probleme ilişkin ağırlık katsayı vektörünün adı yazılabilir. Eğer buraya bir şey yazılmaz ise tüm data değerleri için 1 değeri (default) /alınır.

Şekil 12.25'de **Data set** name ifadesinin sağ tarafındaki boşluğa **x** ve y eksenlerini oluşturan vektörlerin adları (kendiliğinden) yazılır. Kullanıcı dilerse buraya başka adlar da yazabilir.

Eğer x ve y eksenlerini oluşturan data değerleri içinde Inf sayısı var ise ihmal edilir (göz önüne alınmaz), complex sayı var ise reel kısmı alınır. Bu iki durumda da ekranda bir pencere belirir ve kullanıcı uyarılır.

Verilen problemde **MATLAB** arka planında saklı olan iki adet vektör **(cdate** ve **pop)** kullanılmıştır. Kullanıcı isterse bunların yerine **Workspace** ortamında oluşturacağı iki adet (eşit boyutta) vektör kullanarak da benzer işlemleri yapabilir.

12.7.1.2.2. Eğri uydurma (Fitting)

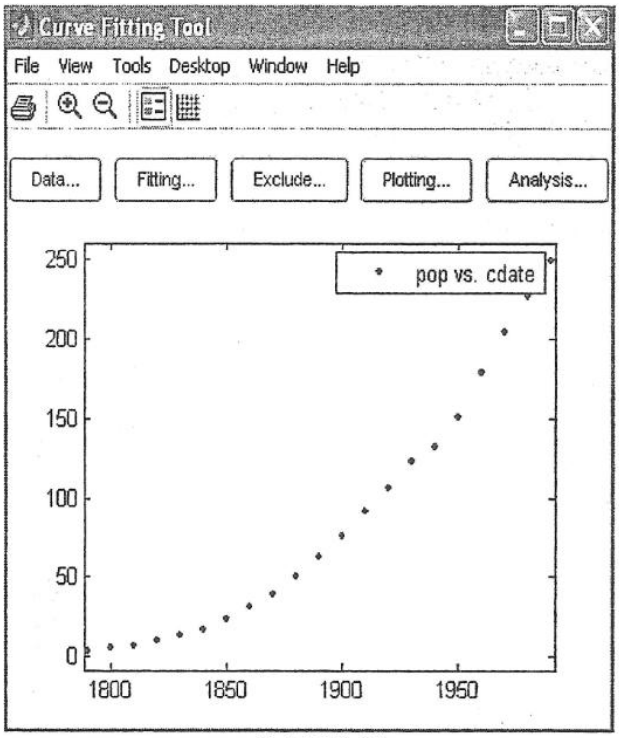
Daha önceki adımlarda x ve y eksenine ilişkin data değerleri cf tool ortamına taşındı. Şekil 12.26'da Fitting... komutuna 'tık'lanıldığmda şekil 12.27'de gösterilen Fitting penceresi ile karşılaşılır. Bu pencerede New Fit komutuna 'tık'lanıldığmda ise şekil 12.28'de verilen görüntü elde edilir (Fitting penceresi). Bu pencerede Fitname komutunun sağ tarafında bulunan boşluğa çizdirilecek eğrinin ismi yazılmalıdır (ör:eğri2). Daha sonra yer alan Polynomial alt penceresinde yer alan eğri türlerinden birisi seçilerek (örneğin; quadratic) Apply ya da Immediate apply komutlarına 'tık'lanıldığmda, Fitting penceresi şekil 12.29 ile verilen forma dönüşür. Bu işleme ilişkin istatistiksel bilgiler ise şekil 12.29'da Results penceresinde yer alır.

Kullanıcı dikkat ederse Curve Fitting Tool penceresi içinde bir adet eğri görecektir. Eğer bu pencerenin altında eğriye ilişkin rezidü değişimi de görülmek isteniyor ise şekil 12.26'da gösterilen Curve Fitting Tool penceresinde yer alan View menüsünün içinde yer alan Residuals/Line Plot seçeneğine 'tık'lanılmalıdır.

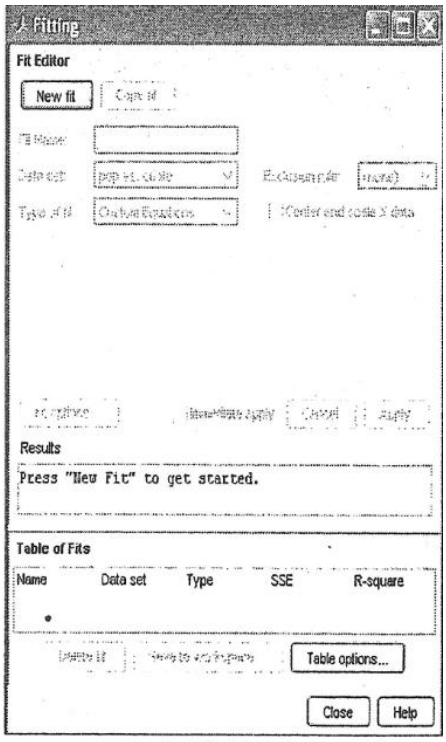
Bu işlem yapıldığında şekil 12.26 yerine şekil 12.30'da verilen pencere elde edilir. Eğer kullanıcı daha yüksek mertebeden eğri uydurmak isterse ekranda;

Equation is badly conditioned. Remove repeated data points or try centering and scaling.

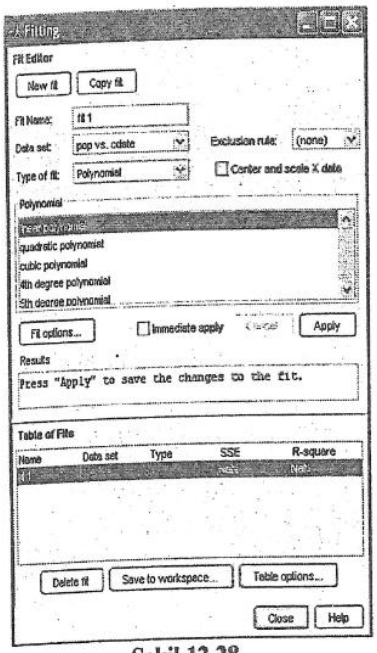
uyarısı ile karşılaşabilir. Bu durumda şekil 12.29'da Center and scale X data ifadesinin sol tarafında yer alan kutucuk işaretlenerek data değerleri normalize edilebilir.



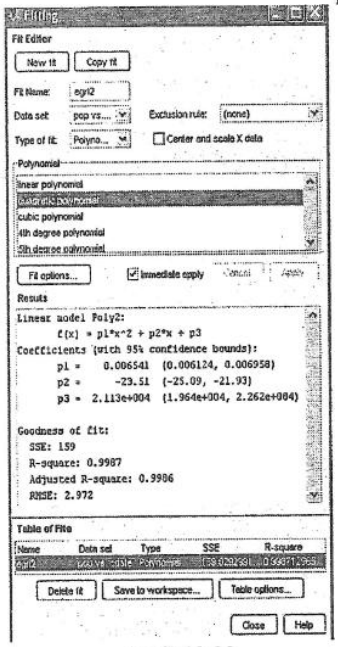
Şekil 12.26



Şekil 12.27

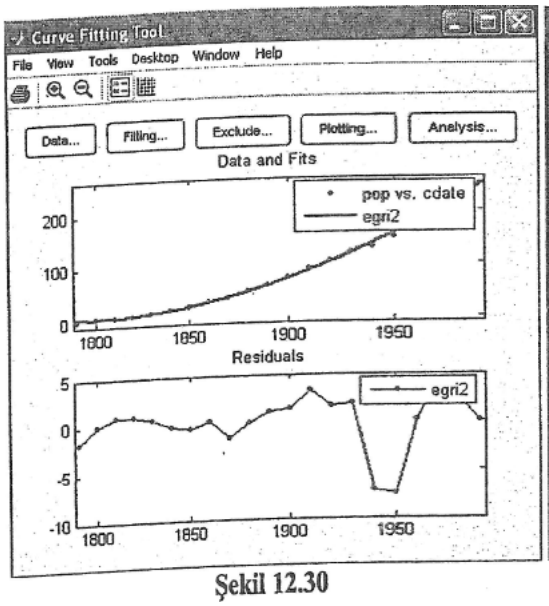


Şekil 12.28

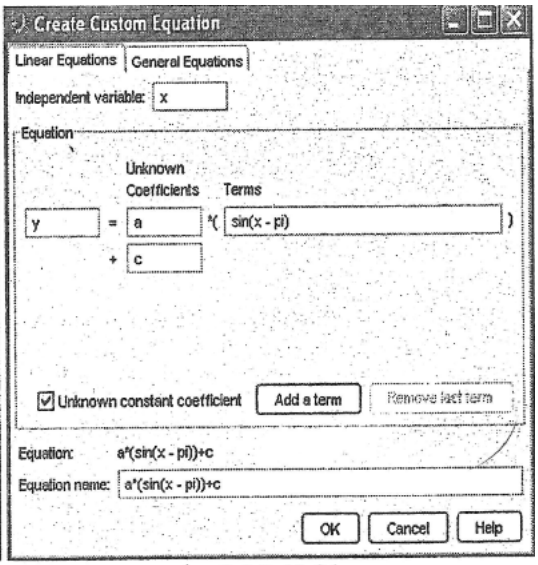


Şekil 12.29

Verilen problem için şekil 12.29'da Center and scale X data işaretlendiğinde, arka planda yapılan normalizasyon işlemi aşağıda verilmiştir:



Şekil 12.30



Şekil 12.31

12.7.1.2.3. Eğri uydurmada işleminde en iyi eğrinin tespit edilmesi

Kullanıcı, yukarıda bahsedilen işlemleri dilediği kadar eğri türü üzerinde deneyebilir. Şekil 12.29'da Type of Fit seçeneğinin sağ tarafındaki boşlukta yer alan ok «anıldığında kullanıcının karşısına bir çok eğri

türü çıkar. Bu seçenekler; Custom equatiorıs(kulIamcı tarafından belirlenecek eğri türleri), Exponential, Fourier, Gaussian, Interpolant, Polyrıomial(polinom), Power, Rational (oransal), Smoothing Spline, Sum of Sin functions, Weibull olarak sıralanır. Bu eğri tipleri aşağıda tanıtılmıştır:

- Exponentials

  $y = ae^{bx}$

  $y = ae^{bx}+ce^{dx}$

- Fourier Series

  $$y = a_0 + \sum_{i=0}^{n} a_i \cos(nwx) + b_i \sin(nwx)$$

- Gaussian

  $$y = a_0 + \sum_{i=0}^{n} a_i e^{\left[-(\frac{x-b_i}{c_i})^2\right]}$$

- Polynomials

  $$y = a_0 + \sum_{i=0}^{n+1} p_i x^{n+1-i}$$

- Power Series

  $y = ax^b$

  $y = a+bx^c$

- Rationals

$$y = \frac{\sum_{i=0}^{n+1} p_i x^{n+1-i}}{x_m + \sum_{i=0}^{n+1} q_i x^{m-i}}$$

- Sum of Sines

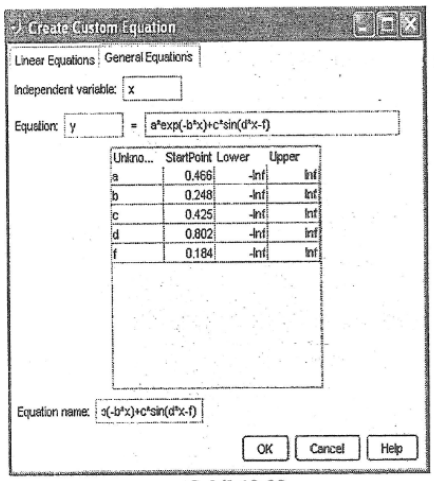
  $$y = \sum_{i=0}^{n} a_i \sin(b_i x + c_i)$$

- Weibull Distribution

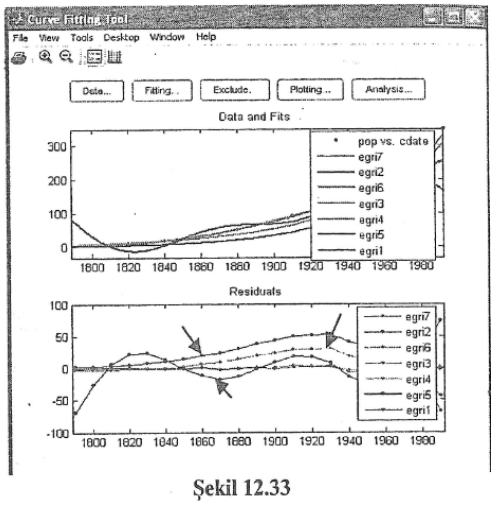
  $y = abx^{b-l}e^{-axb}$

- Custom Equations

Şekil 12.29'da Type of Fit ifadesinin sağ tarafında yer alan boşlukta ok işareti yardımı ile Custom Equations ifadesine ulaşılır. Bu pencerede yer alan New equation.. komutuna 'tık'lanıldığmda şekil 12.31 ile verilen pencere açılır. Bu pencere iki adet seçenek içerir. Bunlardan ilki, Linear Equations diğeri ise General Equations seçeneğidir. Şekil 12.29, Linear Equations'a ilişkindir. Bu pencerede Add a term ifadesine 'tıklayarak y denklemine yeni sinüs eşitlikleri ilave edilebilir yada Remove Last Term komutu kullanılarak y ifadesine en son ilave edilen terim silinebilir. Şekil 12.31'de General Equations seçeneğine 'tık'lanıldığmda şekil 12.32 penceresi ile karşılaşılır. Bu pencerede Equation ifadesinin sağ tarafında bulunan ikinci boşluğa kullanıcı, dilediği eşitliği yazabilir. Bu pencerenin altında yer alan pencerede ise y eşitliğinde kullanılan katsayıların başlangıç değeri, alt ve üst sınır değerlerini girecek pencereler bulunur. Bu değerler girilip, OK tuşuna basılarak eğri denklemi tamamlanır.



Şekil 12.32



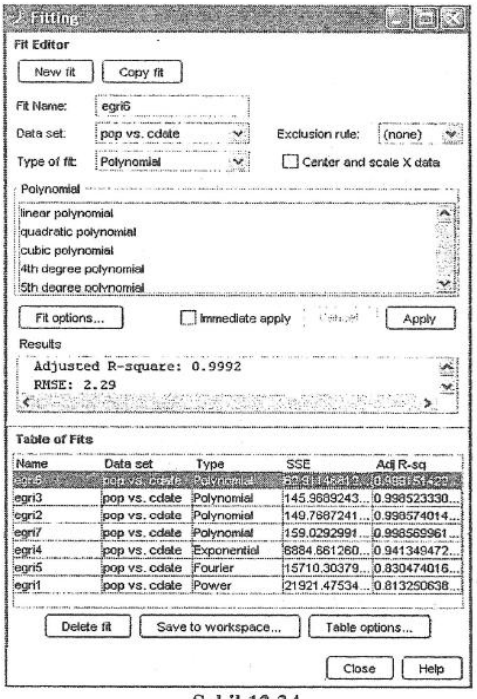
Şekil 12.33

Bu tiplerden hangisi seçilirse Type of Fit ifadesinin altında yer alan boşlukta, seçilen eğri tipine ilişkin eğri denklem (ya da denklemleri) belirir. Kullanıcı, eğri tipini yukarıda bahsedildiği gibi seçtikten sonra şekil 12.29'da görülen Copy Fit komutuna 'tıkladığında açılan (şekil 12.29'a benzeyen) yeni pencerede Fit name ifadesinin sağ tarafına yeni bir isim (ör:egri3) yazmalı ve Immediate apply (ya da Apply) komutunu

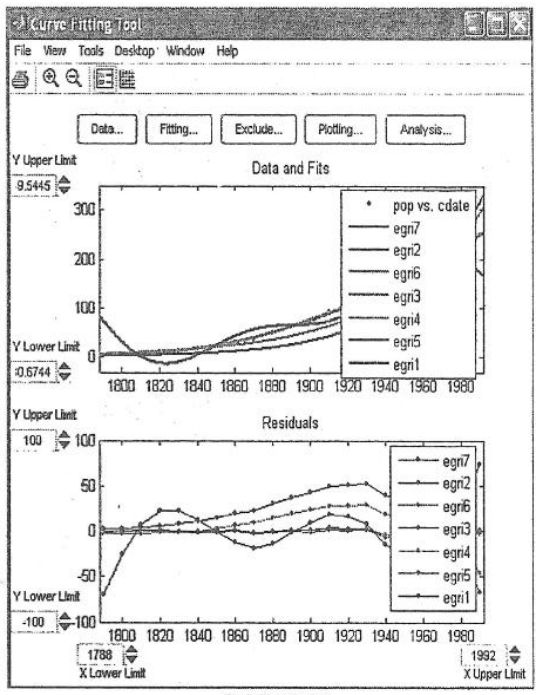
uygulamalıdır. Bu şekilde kullanıcı, dilediği kadar eğri denklemini seçerek eğri uydurma işlemini gerçekleştirebilir. Şekil 12.33'e bakıldığında 7 adet eğri uydurulduğu görülecektir. Şekil 12.34'de en altta yer alan Table of Fits penceresi içinden hangi eğriye 'tık'lanılırsa, aynı şekilde yer alan Results penceresinde bu eğriye ilişkin denklem, katsayı ve istatistiksel değerler yer alır. Table of Fits penceresi yer alan eğriler içinden (isabetli bir eğri uydurma işlemi yapılamadığı düşünülerek) bir tanesi silinmek istenir ise ilgili satır fare ile işaretlenerek Delete fit komutuna 'tık'lanılması gerekir.

Şekil 12.33'de Residuals penceresinde yer alan eğriler içinde birbirlerine benzer olanlar dışında kalan eğriler, kötü bir eğri uydurma işleminin sonucunu göstermektedir. Bu nedenle diğer eğrilerden sapan ve uzaklaşan eğriler şekil 12.34'de Delete fit komutu kullanılarak uzaklaştırılmalıdır. Şekil 12.33'de, bu özelliğe sahip (kötü uydurulmuş 3 adet-egril-egri4-egri5) eğri, ok ile işaretlenmiştir. Kötü olan eğrileri tespit etmek için şekil 12.34'de Table of Fits penceresinde yer alan SSE(Sum of Squares due to Error) ve Adj R-square değerleri de kullanılabilir. Adj R-square değeri l'e yaklaştıkça eğrinin iyi olduğu, l'den uzaklaştıkça ise eğrinin kötü olduğu bilinmelidir. SSE değeri ise küçüldükçe eğrinin iyi olduğu, büyüdükçe eğrinin kötü olduğu söylenebilir (istatistiksel katsayılara ilişkin açıklamalar ilerleyen sayfalarda verilecektir). Bu bilgiler ışığında şekil 12.34'de egril, egri4 ve egri5'e ilişkin SSE ve Adj R-square değerleri incelenmelidir (en iyi eğrinin eğri6 olduğu görülmelidir).

Şekil 12.33'de görülen tüm eğriler bilgisayar ekranında farklı renkte oldukları için kullanıcı tarafından rahatlıkla fark edilebilirler. Kitap siyah-beyaz bash olarak basıldığından bu ayırımı kitap üzerinde görmek zorlaşabilir. Şekil 12.33'de, fare ile herhangi bir eğrinin üzerine gelinip sol tuşa basıldığında, bu noktaya ilişkin x ve y değerlerini gösteren küçük bir pencere açılır.

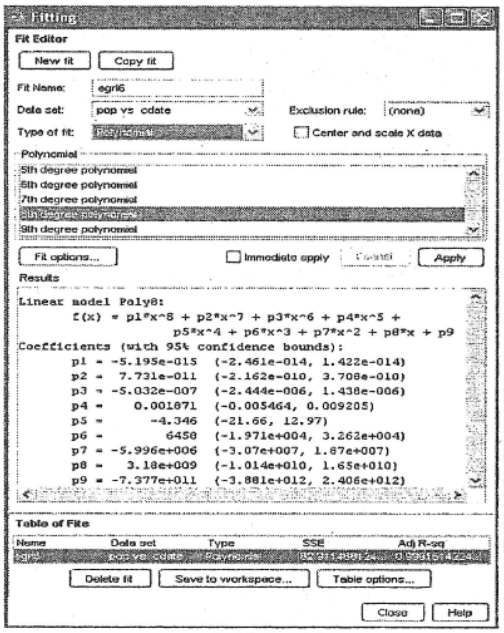


Şekil 12.34

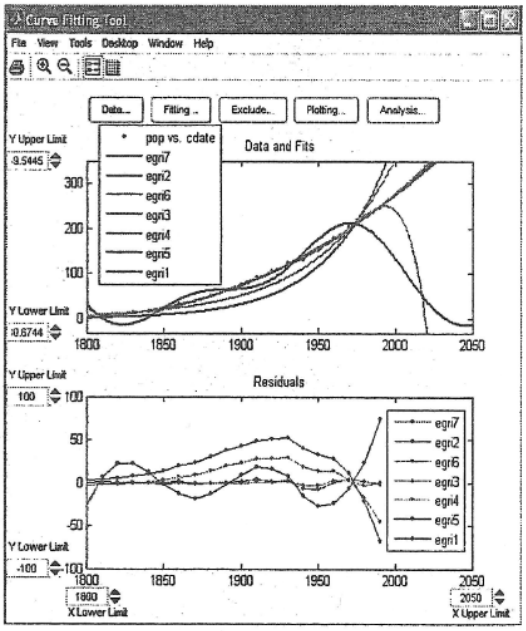


Şekil 12.35

Şekil 12.34'de SSE sütununa fare ile 'tık'lanıldığında bu sütundaki değerler büyüklüklerine bağlı olarak sıralanırlar. Benzer şeyler Adj R-square için de söylenebilir. Eğri 6 değerlerinin sütunun en üstünde olması en iyi eğri uydurmanın bu eğri ile yapıldığını da göstermektedir. Şekil 12.36'da en iyi eğrinin katsayıları gösterilmiştir.



Şekil 12.36



Şekil 12.37

12.7.1.2.4, Dış değer hesabı (Ekstrapoiasyon)

Şimdiye kadar, verilen dataların ait ve üst sınırlan arasında kalan değerleri içeren bir eğri uydurma işlemi gerçekleştirildi. Mevcut data değerlerini kullanarak bu data değerlerinin dışında kalan değerler için de bilgi istenebilir. Örneğin; verilen problemde edat e değerleri; 1790 ile 1990 arasında, pop değerleri ise 3.9 ile 248.7 arasında değişmekteydi. Eğer kullanıcı mevcut 7 adet eğriyi kullanarak 1790'dan küçük ve/veya 1990'dan büyük seneler için pop değerlerini bulmak isterse (extrapolasyon) cftool ortamında ne yapmalıdır?

Yukarıda verilen soruyu cevaplamak için öncelikle şekil 12.33'de Tools menüsünde yer alan Axis Limit Control ifadesine 'tık'lanılmalıdır. Bu işlem yapıldığında kullanıcının karşısına şekil 12.35'de verilen pencere çıkar. Bu pencere iki adet alt pencereden oluşmaktadır; Data and Fits ve Residuals. Bu pencerelerde yatay eksen için X Lower Limit-X Upper Limit ve düşey eksenler için Y Lower Limit ve Y Upper Limit butonları bulunmaktadır. Bu butonların sağ tarafında bulunan aşağı ve yuları ok işaretlerine 'tıklanılarak limit değerleri azaltıp artırmak mümkündür. Örnek olarak X değerleri 1800 ile 2050 aralığında seçilirse şekil 12.36'da gösterilen pencere elde edilir. Bu değişim bir liste olarak görülmek

istenir ise şekil 12.37'de Analysis. . butonuna 'tık'lanılmalıdır. Bu işlem yapıldığında kullanıcının karşısına şekil 12.38'de verilen pencere çıkar. En iyi eğri uydurmanın egri6 ile yapıldığı bilindiği için şekil 12.38'da Fit to analyze ifadesinin sağ tarafındaki boşluğa eğri6 getirilmiş, bu kutunun altında yer alan Analyze at Xi kutusu içine ise istenilen aralık ve artış miktarı yazılmalıdır. Şekil 12.38'deki uygun kutucuklar işaretlenip Apply seçeneğine 'tık'lanıldığında şekil 12.38'de sağ tarafta yer alan (1800 ile 2050 aralığındaki) edat e ve (bu aralığa karşı gelen) pop değerler listesi elde edilir.

Şekil 12.38'de Save to workspace.. butonuna basıldığında kullanıcının karşısına küçük bir pencere çıkar. Bu pencere içindeki boşluğa bir isim yazıldığında, Workspace ortamında bu adla anılan bir yapı dosyası içine ekstrapoiasyon değerleri kaydedilir.

12.7.1.2.5. Table of fits seçenekleri

Şekil 12.34'de Table of fits penceresinde uydurulan eğriler hakkında bilgi vermek üzere yerleştirilmiş iki adet istatistik tanımı yer almaktadır: SSE ve Adj R-square. SSE değeri en az karesel hata hesabını yapan bir istatistiksel büyüklüktür. Bu değer sıfıra doğru yaklaştıkça, iyi bir eğri uydurma işleminin gerçekleştirildiği düşünülebilir. Eğri uydurma işleminde diğer bir kalite ölçüsü ise Adjusted R-square katsayısıdır. Bu değerin, iyi bir eğri uydurma işleminde l'e yaklaşması istenir. Kullanıcının Table of fits seçeneklerini değiştirmesi, diğer bir ifade ile istatistiksel ölçme büyüklüklerini belirlemesi de mümkündür. Bunun için şekil 12.34'de yer alan Table options.. butonuna 'tık'lanmalıdır. Bu işlem yapıldığında şekil 12.39'da gösterilen pencere ile karşılaşılır. Bu pencerede yer alan kutucuklar işaretlendiğinde, bunların sağ tarafında yer alan ifadeler, Table of fits içinde gözükür. Şekil 12.34'de görülen Table of Fits ayarları, şekil 12.39 yardımı ile gerçekleştirilmiştir. Şekil 12.39'da görülen istatistiksel tanımlar aşağıda kısaca tanıtılmıştır:

-SSE (The sum of squares due to error)

yi değeri ölçülen data vektörünün i. elemanım, y_i eğri uydurma sonunda elde edilen data vektörünün i. elemanını, w_i ise ağırlık katsayılarını göstermek üzere;

$$SSE = \sum_{i=1}^{n} w_i (y_i - \overline{y_i})^2$$

eşitliği ile hesaplanır. Bu değer, uydurulan değerlerin sapma miktarım ölçer. SSE, O'a yaklaştıkça eğri uydurma işleminin başarılı olduğu söylenebilir.

-R-square

y vektörü ile, y vektörü arasındaki korelasyonun karesine eşit bir katsayıdır. İki büyüklüğün birbirine bölümünden elde edilir (n; y (veya y) vektörü eleman sayısı):

$$R=\frac{SSR}{SST}=\frac{\sum_{i=1}^{n} w_i(\bar{y}_i-\bar{y})^2}{\sum_{i=1}^{n} w_i(y_i-\bar{y})^2}=1-\frac{SSE}{SST};\ SST=SSR+SSE$$

Çoklu saptama katsayısı olarak isimlendirilebilir. Bu değer, eğri uydurmanın data dağılımındaki başarısını ölçer. R değeri l'e yaklaştıkça, eğri uydurma işleminin başarılı olduğu söylenebilir.

-Adj R-sq (Adjusted R-square)

v (değeri büyük olması istenir) rezidü serbestlik derecesi olmak üzere, Adj R-sq; aşağıdaki ifade ile hesaplanır.

$$\text{Adj R-sq}=1-\frac{SSE}{SST}\frac{n-1}{v-1}$$

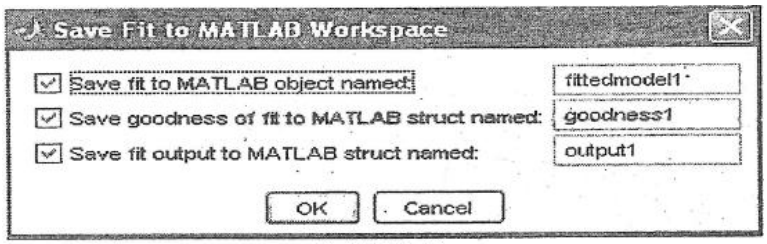
Modele ilave katsayılar ilave edildiğinde, genellikle Adj R-sq, eğri uydurmada en iyi verimlilik göstergesidir, l'e yakın olması istenir.

-RMSE (Root Mean Squared Error)

Ortalama karesel hatanın (MSE)kareköküdür. 0'a yakın olması istenir.

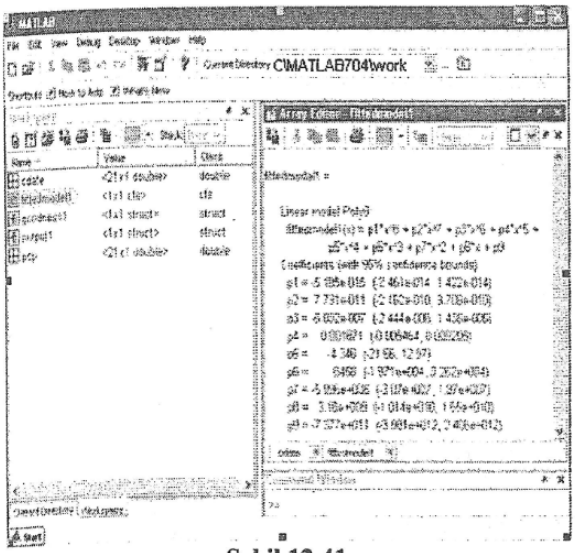
$$RMSE=\sqrt{MSE}=SSE/v$$

12.7.1.2.6. Eğri uydurma sonuçlarının kaydedilmesi

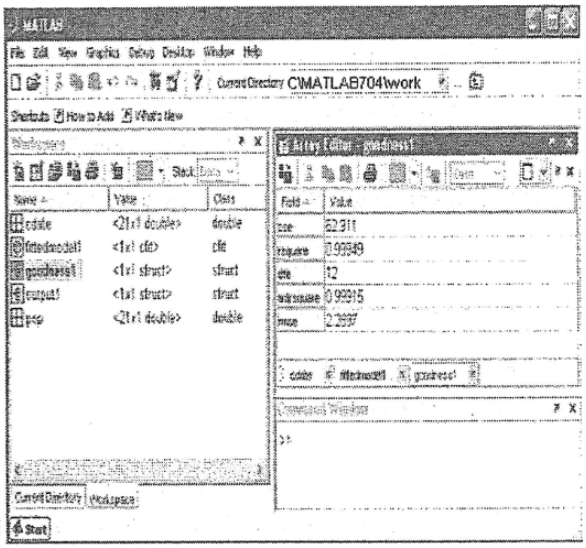


Şekil 12.40

Şekil 12.34'de Save to Workspace butonuna 'tık'Ianıldığında, şekil 12.40'da verilen pencere açılır. Bu pencerede üç adet küçük kutucuk yanında üç adet ifade yer almaktadır. Örneğin en üstte yer alan Save fit to MATLAB object named ifadesinin sol tarafında yer alan kutucuk fare yardımı ile 'tık'Ianıldığında, sağ tarafta yer alan boş dikdörtgen kutu içinde yazılı olan fittedmodell adlı cfit objesi, Workspace ortamında oluşacaktır. Şekil 12.41'de Workspace ortamında yüklenen fittedmodell üzerine çift adet 'tık'Ianıldığında en iyi eğri uydurma modeline ilişkin polinom katsayıları, Array Editor ortamında belirir. Benzer işlem goodnessl yapısı için yapılırsa, en iyi eğriye ilişkin istatistiksel değerler, Array Editor ortamında, şekil 12.42'de gösterildiği gibi ortaya çıkar. Outputl yapısı içinde ise kullanılan yaklaşım ile ilgili ilave açıklamalar yer alır.

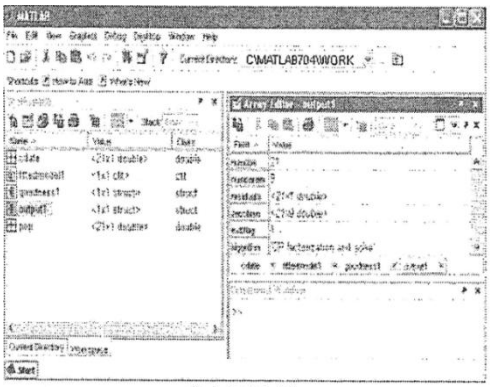


Şekil 12.41

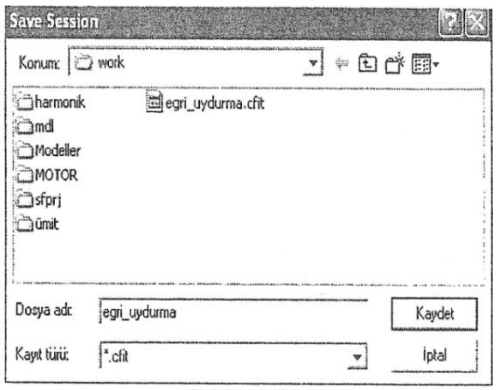


Şekil 12.42

Kullanıcı şekil 12.34'de görülen eğri türlerini ileride kullanılmak üzere saklamak isteyebilir. Bunun için önce (Curve Fitting Tool) şekil 12.37'de File menüsü içinde yer alan Save Session.. komutu 'tık'lanılmalıdır. Bu işlem yapıldığında kullanıcının karşısına şekil 12.44'de verilen pencere çıkar. Kullanıcı tarafından bir dosya adı altında (ör:egri_uydurma) tüm eğriler, Kaydet butonuna 'tıklanılarak kaydedilir. Bu tür dosyaların uzantısı cfit olur. Eğer MATLAB ortamından çıkılır (veya bilgisayar kapatılır) ve daha sonra tekrar bu eğrilere ulaşılmak istenirse, şekil 12.37'de gösterilen Curve Fitting Tool penceresinde File menüsü altmda yer alan Load Session.. komutuna 'tıklanılır. Bu işlem yapıldığında şekil 12.44'e benzer bir pencere açılır. Burada egri_uydurma. cfit seçilir ve Aç butonuna 'tıklanılır ise şekil 12.37'deki pencere açılır.

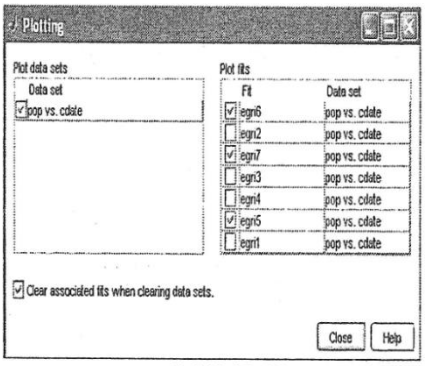


Şekil 12.43

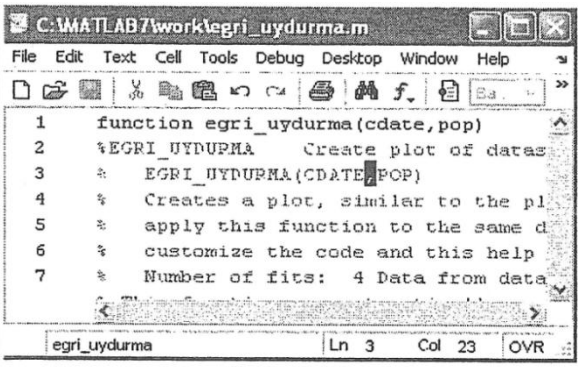


Şekil 12.44

Eğer kullanıcı 7 adet eğriden bazılarını silmek istiyor ise şekil 12.37'de Plotting. . butonuna 'tıklamalıdır. Bu işlem yapıldığında şekil 12.45'de gösterilen pencere çıkar. Bu pencerede silinmek istenen eğrilerin isimlerinin sol tarafındaki kutucuklar boş bırakılır ve pencerenin sol altındaki Clear.. ifadesinin yanındaki kutucuk işaretlenir, Close butonuna 'tıklanılır. Bu işlem sonunda şekil 12.37'de görülen egril, egri2, egri3 ve egri4 eğrileri ortadan kalkar.



Şekil 12.45



Şekil 12.46

Kullanıcı isterse yukarıda bulunan 7 adet eğriyi M dosyası (altprogram olarak) saklayabilir. Bu işlemi yapmak için şekil 12.37'de verilen

Curve Fitting Tool penceresinde File menüsü içinde yer alan Generate M-File ifadesine 'tıklanılır. Bu işlem sonunda şekil 12.44'e benzer bir pencere açılır. Bu M dosyasına, uygun bir isim verilerek (ör: egri_uygurma .m), Kaydet butonuna 'tık'lanılır.

Eğer kullanıcı, yukarıda belirtildiği şekilde kaydedilen egri_uydurma.m adlı altprogramı, bir MATLAB dosyası olarak çalıştırmak isteyebilir. Bunun için şekil 12.46'da bir kısmı gösterilen egri_uygurma. m adlı altprogram dosyasının ilk satırına bakılmalıdır. Şekil 12.46'da ilk satır;

function egri_uydurma (cdate, pop)

olarak verilmektedir. Bu satırda function komutundan sonra gelen 'egri_uydurma{cdate,pop)' ifadesi, Command Window ortamında aşağıdaki;

»egri_uydurma (cdate, pop) (‘enter’)

şekilde uygulanırsa, egri__uygurma.m adlı altprogram çalışır ve sonuçta şekil 12.37'de görülen eğriler elde edilir. Yukarıda verilen satırın çalışabilmesi için cdate ve pop vektörlerinin Workspace ortamında tanımlı olması gerekir.

12.7.1.2.7. Data değerleri içindeki gürültünün süzülmesi (smooting)

Eğer verilen data değerleri gürültü içeriyor ise kullanıcının bu değerleri ortadan kaldırmak için bir süzme algoritmasına ihtiyacı olacaktır. Eğri uydurma işlemi parametrik bir işlem olmasına rağmen süzme, parametrik olmayan bir uydurma işlemidir. Süzme işlemi, gürültü içeren data değerlerini data vektörü içinden uzaklaştırma işlemi olarak da yorumlanabilir. Süzme işleminde yaygın olarak kullanılan iki yaklaşım vardır: filtreleme ve regresyon. Her iki yaklaşım da bir aralığa span (pencere) ihtiyaç duyar. Bu pencere işlem yapılan data değerleri ile beraber yer değiştirir. Span büyüdükçe eğrideki pürüzlülük azalır fakat pürüzlüğü alman datanın çözünürlüğü azalır, span küçüldükçe tam ters sonuçlar elde edilir. Optimal

span değeri, data değerlerine ve kullanılan süzme yöntemine bağlıdır. Deneyerek bulmak en yaygın yaklaşımdır.

Curve fitting toolbox (eğri uydurma araç kutusu) başlıca aşağıda verilen süzme yöntemlerini kullanır.

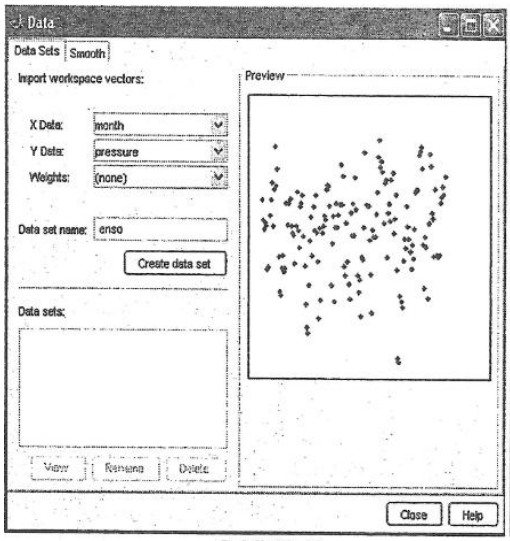
- Moving average filtering-alçak geçiren bir filtredir, komşu noktaların ortalamasını alır.
- Lowess ve loess-Lineer en küçük kareler yöntemi ile eğri uydurur,birinci dereceden polinom (lowess)veya ikinci dereceden polinom (loess) kullanır.
- Savitzky-Golay fıltering-ilk yönteme benzer, filtre katsayılarının türevini alır ve çeşitli derecelerden polinomları kullanır.

Kullanıcı isterse düzleştirme için cubic spline komutunu da kullanabilir.

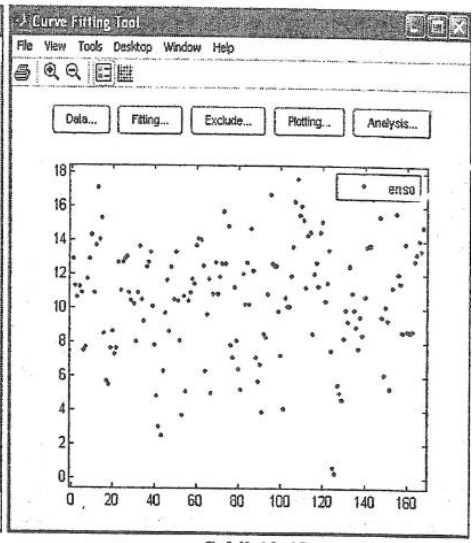
Düzleştirme işlemini iyi açıklayabilmek için MATLAB arka planında saklı bulunan bir başka data çifti kullanılacaktır, cftool ortamında düzleştirme (smooting) işlemi için kullanılan arayüz şekil 12.47'de gösterilmiştir. Bu dataya ulaşmak için;

»load enso ('enter')

komutu uygulanmalıdır. Bu işlem yapıldığında Workspace ortamında, month (ay) ve pressure (basınç) adlı eşit boyutta iki vektör oluşur, y eksenini oluşturan pressure vektörü; aylık olarak Avusturalya'daki iki kritik bölge arasındaki atmosferik basınç ortalamasını içerir (bu bilgi kullanılarak güney yarım küredeki rüzgar bilgilerine ulaşılır). x eksen bilgileri ise aylar içindeki nispi zamanı gösterir. Her bir vektör 168 sayı içerir. Şekil 12.47'de Create data set butonuna 'tık'lanıldığında şekil 12.48'de gösterilen Curve Fitting Tool penceresi açılır. Şekil 14.48'de fare ile herhagi bir nokta üzerine gelinip farenin sağ tuşuna basılır ve ortaya çıkan pencereden Line Style/Solid komutuna 'tıklanılır ise şekil 12.49'da gösterilen eğri değişimi elde edilir. Şekil 12.47'de Smooth butonuna 'tık'lanıldığında ortaya çıkan pencerede Create smoothed data set butonuna 'tık'lanıldığında şekil 12.50'de verilen pencere elde edilir. Bu pencerede Method ifadesinin sağ tarafındaki boşluktaki ok işaretine 'tık'lanıldığında kullanıcının karşısına;



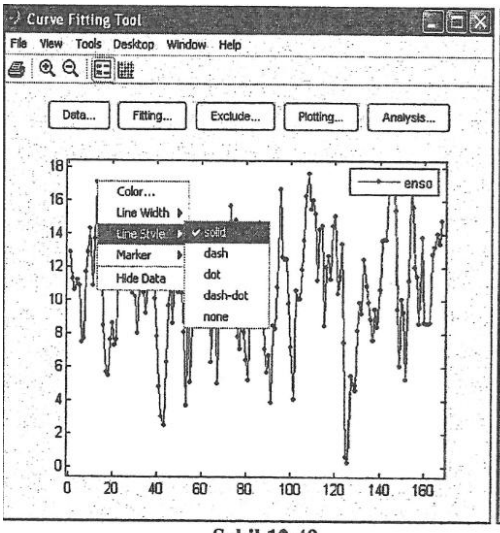
Şekil 12.47



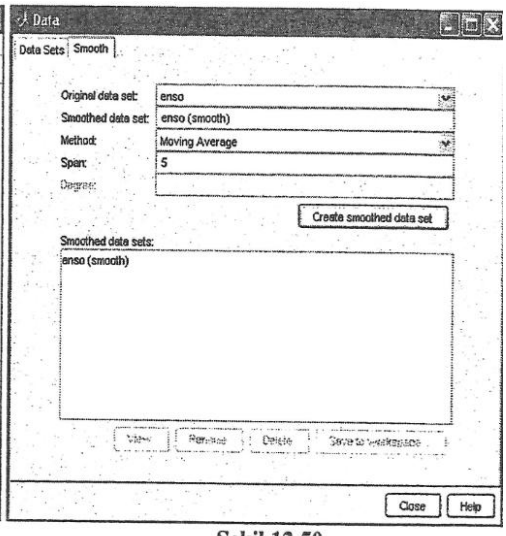
Şekil 12.48

- Moving Average
- Lowess(linear fit)
- Loess (quadratic fit)
- Savitzky-Golay
- Robust Lowess (linear fit)-sınır dışındakilere direnç gösterir
- Robust Loess (quadratic fit)-sınır dışındakilere direnç gösterir

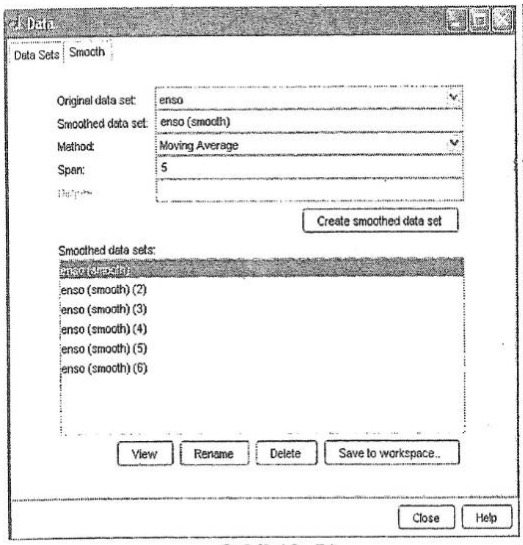
metodları çıkar. Şekil 12.50'de Span ifadesinin sağ tarafındaki boşluğa; her bir düzleştirme hesaplamasında kullanılacak nokta (data) sayısı yazılır. Eğer metod olarak Savitzky-Golay seçilirse Span altındaki boşlukta Degree ifadesi belirir. Buraya polinom derecesi yazılır ve bu değerin Span değerinden az olması istenir.



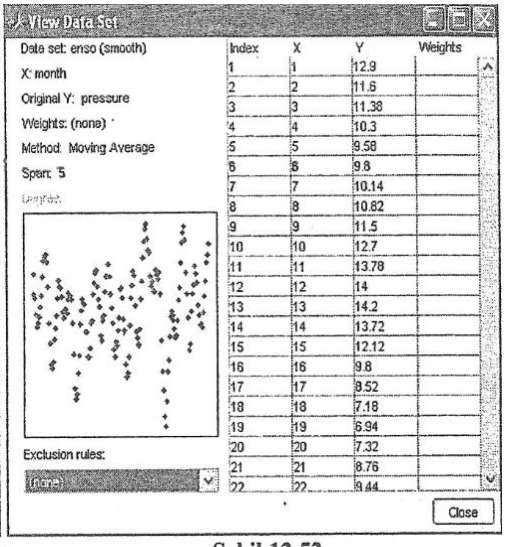
Şekil 12.49



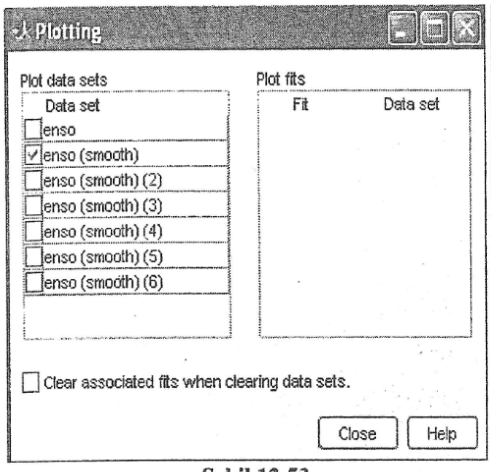
Şekil 12.50



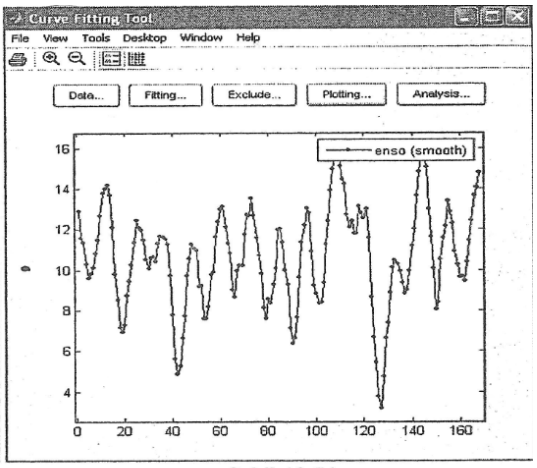
Şekil 12. 51



Şekil 12.52



Şekil 12.53



Şekil 12.54

Enso adlı data dosyasına yukarıda belirtilen tüm metotlara göre ayrı ayrı düzleştirme işlemi uygulanabilir. Bunun için şekil 12.50'den bir metot seçilmeli ve daha sonra pencerede Create smoothed data set butonuna 'tık'lanılmalıdır. Aynı işlem diğer metotları ayrı ayrı seçerek tekrarlanırsa şekil 12.51'de gösterilen pencere elde edilir. Bu pencere içinde yer alan smoothed data set penceresi içinde yer alan çözüm yaklaşımlarından herhangi birine (örneğin:Moving Average) fare ile 'tıklanılıp daha sonra View butonuna 'tıklanılırsa şekil 12.52'de görülen View Data Set penceresi elde edilir (enso (smooth) adı altında). Eğer şekil 12.5l'da Save to workspace butonuna 'tık'lanıldığmda bir pencere açılır. Bu pencere içindeki boşluğa bir ad yazılırsa şekil 12.52'de Index penceresinde görülen data değerleri bu ad altında workspace ortamında bir yapı dosyasında saklanır.

Şekil 12.48'de Plotting. . butonuna 'tıklanılır ise şekil 12.53'de görülen pencere açılır. Bu pencerede yer alan her adın bir düzleştirme metoduna karşı geldiği bilinmektedir. Eğer bunların içinden bir tanesi seçilir ve kutu içine 'tıklanılır ise şekil 12.54'de verilen değişim elde edilir. Eğer kullanıcının ekranında yalnızca

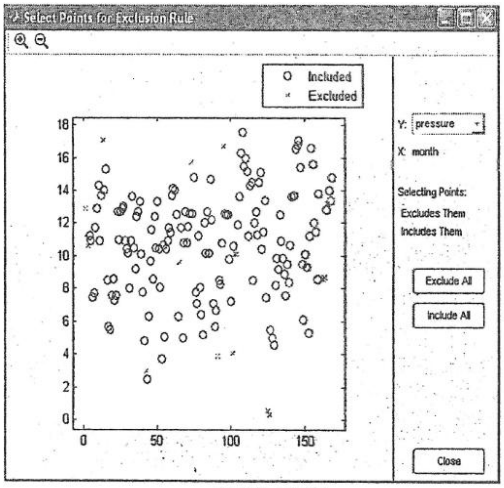
noktalar var ise herhangi bir noktaya farenin sağ tuşu ile 'tık'lanıldığmda açılan küçük pencerede Line S tyle/Sol id komutuna 'tıklamalıdır. Yapılan işlemin sonucunu görmek için şekil 12.49 ile şekil 12.54 birbirleri ile karşılaştırılmalıdır.

12.7.1.2.8. Data içindeki bazı değerlerin eiiminasyonu

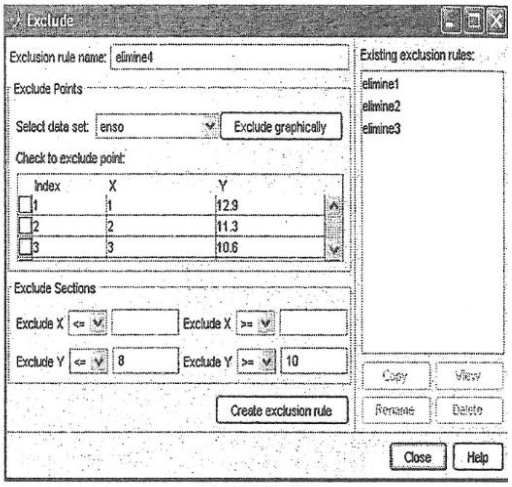
Yukarıda verilen month veya pres sure adlı vektörlerin içinden bazı değerlerin (uygun olmadıkları için) silinmesi gerekebilir. Silinecek data değerleri vektör içinde farklı yerlerde olabilecekleri gibi (örneğin; l.,8. ve 10. elemanlar gibi) birbirini takip eden (bölgesel-örneğin; 18. eleman ile 35. eleman arasındaki

tüm sayılar) sırada da olabilirler, cf tool ortamında her iki durum için de eliminasyon yapılabilir.Bu işlemlerin yapılacağı ara yüze ulaşmak için Curve Fittiıg Tool penceresinde (şekil 12.49) Exc"hıde. . butonuna basılmalıdır. Bu işlem yapıldığında kullanıcının karşısına şekil 12.55'de verilen pencere çıkar. Şekil 12.55'de Select data set ifadesinin sağ tarafındaki boşlukta yer alan ok işaretine 'tıklanılarak enso adlı (içinde iki vektör içeren) dosya işaretlenmelidir.

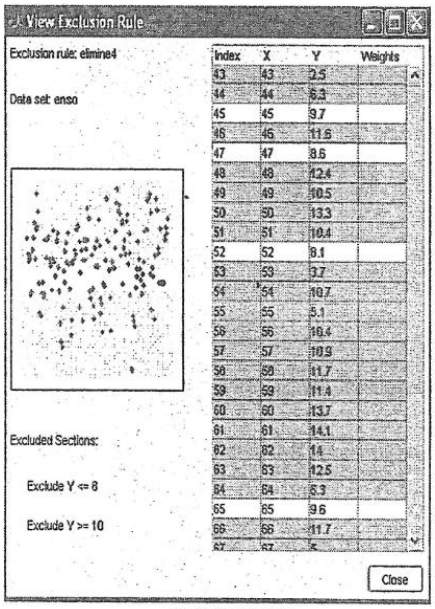
Böylece eliminasyon işleminin bu dosyada yer alan vektörler içinde yapılacağı belirlenmiş olmaktadır. Eğer daha önce bir ad altında eliminasyon işlemi gerçekleştirilmiş ise (şekil 12.56) bunların isimleri Existing exlusion rule penceresi içinde yer alır (eliminel ve elimine2 gibi). Eğer yeni bir ad altında (ör:elimine3) bir eliminasyon işlemi daha gerçekleştirilmek isteniyor ise şekil 12.56'de gösterilen Exclusion rule name ifadesinin sağ tarafındaki boşluğa bu isim yazılmalıdır. Eğer enso içinde **X** değerlerinden bazıları (tek-tek) elimine edilecek ise Check to exclude enso alt penceresi içindeki satırların sol başındaki kutucuklara fare yardımı ile 'tıklanılması gerekir. Şekil 12.56'de **X** (veya **Y)** eksenindeki ilk üç değer, elimine3 adı altında enso dosyasından (diğer bir ifade ile month ve pressure vektörlerinin i l k ü ç elemanı bu vektörler içinden) uzaklaştırılmaktadır. Bu işlemin uygulamaya sokulabilmesi için son olarak Create exclusion rule butonuna 'tıklanılması gerekir. Kullanıcı isterse elimine ettiği elemanları grafik ortamında da görüntüleyebilir. Örneğin, daha önce gerçekleştirilmiş olan elimine3 adlı dosya elemanlarını grafik ortamda görmek istersek şekil 12.56'de elimine2 adlı dosya üzerine 'tıklandıktan sonra Exclude graphically butonuna 'tıklanılmalıdır. Bu işlem yapıldığında kullanıcının karşısına şekil 12.57'de görülen pencere çıkar. Burada 'x' işareti ile gösterilen değerler pressure vektöründen elimine edilen değerlerdir, 'o' işareti ile gösterilen değerler ise yeni pressure vektörünü oluşturan değerlerdir.



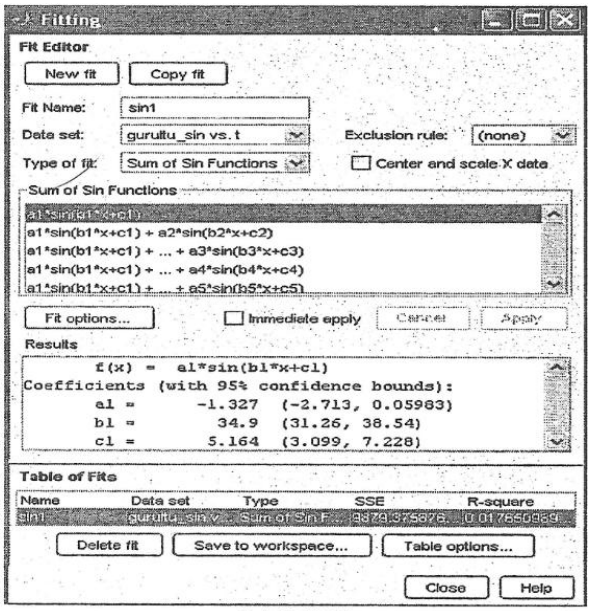
Şekil 12.57



Şekil 12.58



Şekil 12.59



Şekil 12.60

Şekil 12.57'de Exclude All butonuna 'tık'lanıldığmda buradaki tüm değerler pressure vektöründen uzaklaştırılırken, Include All butonuna basıldığında ise elimine edilen tüm değerler geri alınır.

Şekil 12.56'da Exclude Section penceresinde ise elimine edilecek bölgeler belirlenir. Örneğin Y vektörünün 8'e eşit ve bu değerden küçük değerleri ile Y değerlerinin 10'a eşit ve değerden büyük olduğu bölgeler, enso içinden uzaklaştırılmak istensin. Bunun için ilk adım olarak şekil 12.58'de gösterildiği gibi boşluklar doldurulmalıdır. Create exclusion rule butonuna 'tık'lanıldığmda bu bölge elimine4 adı altında kaydedilir. Eğer Exclude graphically butonuna 'tık'lanılırsa şekil 12.59'da gösterilen pencereye ulaşılır. Bu pencerede sol tarafta elimine edilen bölgeler gri tonla, geri kalan (elimine edilmeyen) bölgeler ise açık tonla işaretlenmiştir. Sağ tarafta yer alan tabloda ise gri satırlar elimine edilen değerleri, açık renkli satırlar ise elimine edilmeyen değerleri gösterir (bu tabloda tüm değerler gözükmemektedir). Şekil

12.58'de sağ alt tarafta yer alan Copy, View, Rename ve Delete butonları aktif değildir. Bu bütanların aktif olabilmesi için Existing exlusion rule penceresi içinde yer alan adlardan bir tanesine 'tık'lamak gerekir. Bu butonlar kullanılarak bu pencere içinde yer alan dosyalar üzerinde kopyalama, çizdirme, isimlendirme ve silme işlemleri yapılabilir. Eğer şekil 12.57de 'o' işaretlerinin üzerine fare ile gelinip sol tuşa basılırsa, bu işaret V işaretine dönüşür (Excluded olur). Eğer farenin sol tuşu tutularak işaretler seçilir ise tuş bırakıldığında seçilen 'o' işaretleri 'x' işaretine dönüşür.

12.7.1.2.8.l. Bölgesel eliminasyon örneği

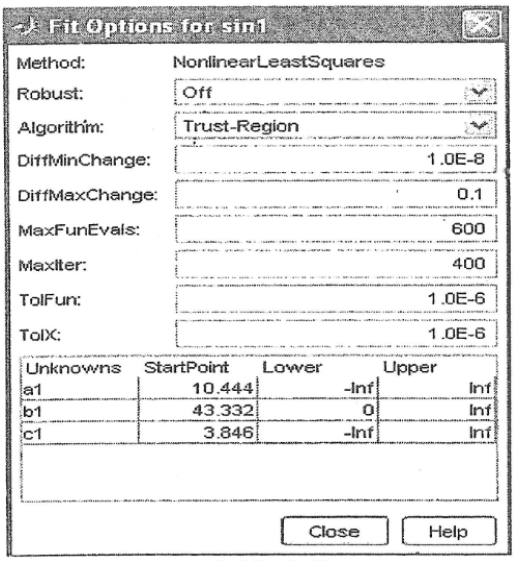
Yukarıda data eliminasyon işleminin iki türlü; tek-tek ya da bölgesel olabileceği belirtilmişti. Bilindiği gibi tüm parametrik eşitliklerde, Curve Fitting Toolbox başlangıç katsayılarını şart koşmaktadır. Bazı tip datada (örneğin; bir çok periyot içeren data) iyi bir başlangıç değeri seçilemez ise tatminkar bir sonuç alınamaz. Böyle durumlarda bölgesel eliminasyon, uygun bir başlangıç değeri tespit edilme işleminde kullanıcıya yardımcı olabilir. Burada bölgesel eliminasyona örnek olarak gürültü içeren bir sinüs dalgasında başlangıç değer kestirimi yapılacaktır. Aşağıdaki MATLAB satırlarında gurultu_sin adı ile

içinde gürültü barındıran bir sinüzoidal dalga üretilmektedir. Bu dalganın üç önemli parametresi (sırası ile); genlik=10, açısal frekans=16*pi ve faz farkı=pi/4 olarak, param adlı 3 elemandan oluşan vektörde saklanmaktadır. Zaman ekseni olarak seçilen t vektörü ; 0 ila 1 saniye arasında değişmektedir, artış değeri olarak 0.005 sn alınmaktadır. Gürültü işaretinin üretilmesinde (her zamanki gibi) rand komutu kullanılmaktadır.

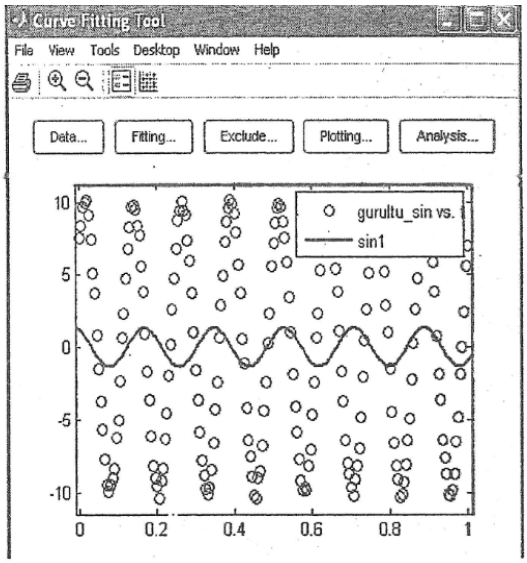
```
»rand (‘state’,0)                    ('enter')
»t= [0:0. 005:1] ;                   ('enter')
»param= [10 16*pi pi/4];             ('enter')
»gurultu_sin =param(l) * (sin{param(2) *t+param(3) }) + (rand(size(t)) -0 .5} ;
                                     ('enter')
```

Yukarıdaki satırlar uygulandığında X ekseni olarak t vektörü, Y ekseni olarak ise gurultu_sin değerleri kullanılacaktır. Daha önce de açıklandığı gibi; cftool komutu uygulandığında açılan Curve fitting Tool penceresinde Data. . butonu yardımı ile bu iki vektör X Data ve Y Data olarak atanırlar. Daha sonra Curve fitting Tool penceresinde Fitting. . butonuna 'tık'lanıldığmda açılan pencerede New Fit butonuna 'tık'lanılmalıdır. Bu işlem sonunda gerekli ayarlamalar yapıldığında şekil 12.60'da verilen görüntü elde edilir. Bu pencerede Data set ifadesinin sağ tarafındaki boşluğa (üretilen eğri sinüs formunda olduğu için) sini yazılmıştır. Type of f i t ifadesinin sağ tarafında ise Sum of Sin Functions seçeneği bırakılmalıdır. Bu pencerede Fit options, .tuşuna basılırsa şekil 12.61'de verilen görüntü, Apply yanındaki kutucuk 'tıklanırsa şekil 12.62 elde edilir. Şekil 12.61'de S t ar t Point değerleri incelendiğinde al (sinüs işaretinin genlik değeri hariç) diğerlerinin param değerlerine yalan olmadığı görülecektir. Şimdi şu soru sorulabilir: S tart Point değerleri, param değerlerine nasıl yaklaştırılabilir? Bu işlem 3 adımda gerçekleştirilebilir. Bu adımlar sıra ile aşağıda gösterilmiştir:

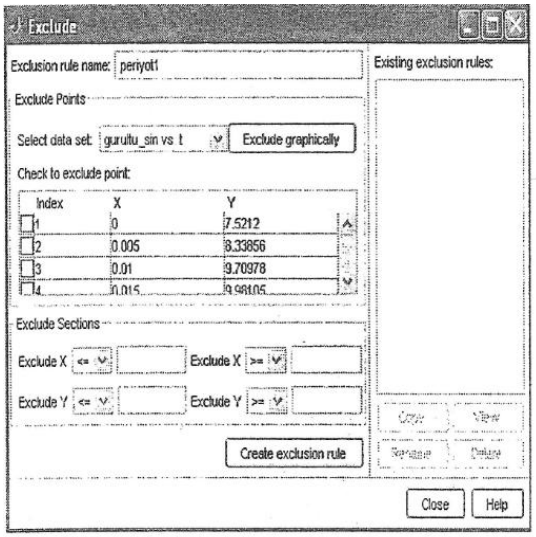
1. Şekil 12.62'de Exclude. . butonuna 'tık'lanılmalıdır. Bu işlem yapıldığında şekil 12.63'de gösterilen pencere elde edilir. Bu pencerede Exclusion rule name ifadesinin sağ tarafındaki boşluğa (yalnızca 1 periyottuk data değerleri inceleneceği için) periyot1 yazılmıştır. Select data set ifadesinin sağ tarafına ise oku kullanarak gurultu_sin vs t ifadesi getirilmelidir. Şekil 12.63'de



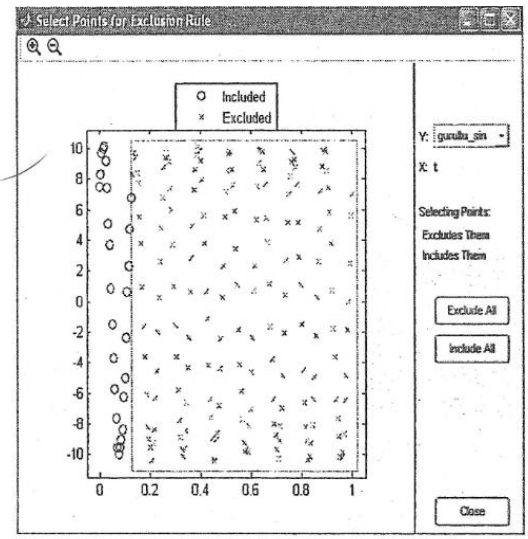
Şekil 12.61



Şekil 12.62



Şekil 12.63

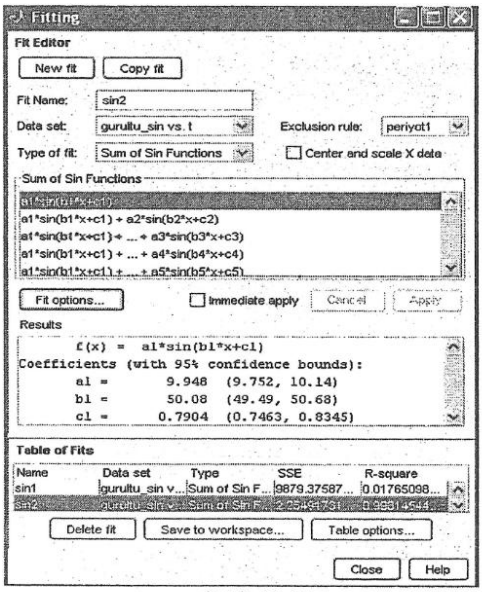


Şekil 12.64

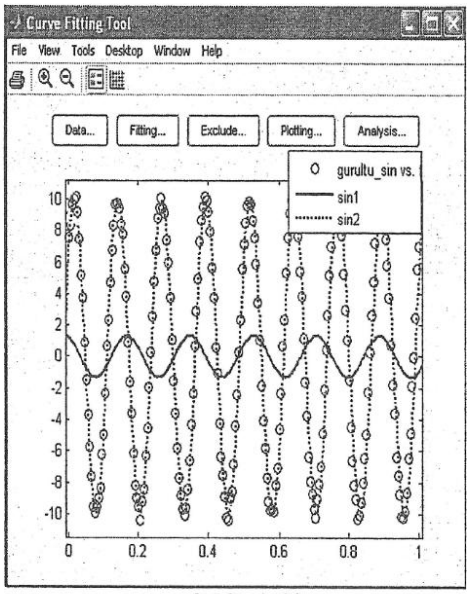
Exclude graphically butonuna 'tık'lanıldığmda Select Points for Exclusion Rule penceresi açılır. Bu pencerede farenin sol tuşu (basılı tutularak) yardımı ile 1 periyotluk bir zaman (0.125 saniye) dilimi dışında kalan bölge işaretlenir ve sol tuşu bırakılırsa şekil 12.64'de verilen pencere elde edilir. Bu pencerede 'x' işaretli data değerleri, Y eksenini oluşturan gurultu_sin adlı vektörün dışına çıkarılmıştır. Şekil 12.63'de Create exclusion rule butonuna 'tık'lanıldığmda, şekil 12.64'de 'o' işareti ile gösterilen data değerleri periyot1 adlı dosya içine kaydedilmiş olmaktadır.

2- Şekil 12.62'de Fitting.. butonuna 'tık'lanıldığmda açılan Fitting penceresinde Copy fit komutuna 'tık'lanıldığmda yeni bir Fitting penceresi ortaya çıkacaktır. Bu pencerede Fitname boşluğuna sin2, Type of fit boşluğuna Sum of Sin Functions yazılarak, Exclusion rule boşluğuna ise ok yardımı ile periyot1 ifadesi getirildiğinde şekil 12.65 elde edilir. Bu pencerede Apply. .butonuna 'tık'lanıldığmda ise şekil 12.66'da verilen pencere elde edilir. Şekil 12.66 incelendiğinde genel olarak kabul edilebilir bir 'eğri uydurma yaklaşımı'na ulaşıldığı söylenebilir.

3- Son adımda ise, 2. adımda ulaşılan al,a2 ve a3 katsayıları kullanılarak ile 1. adımdaki işlemler tekrar yapılacaktır (eliminasyon yok). Bunun için ilk adım olarak şekil 12.65'de Table of Fits içindeki sini ifadesi üzerine fare ile 'tık'lanılmalıdır. Daha sonra Fit options. . tuşuna 'tık'lanılmalı ve açılan pencere içindeki al=10.444, a2=43.332 ve a3=3.646 katsayıları yerine 2. adımdaki al=9.948, a2=50.Q8 ve a3=0.7904 katsayıları yazılmalıdır (şekil 12.67). Yeni koşullarda şekil 12.65'de Fit options, .butonuna 'tık'lamldığmda şekil 12.68'in üst penceresi elde edilir. Bu pencerede View/Residuals/Line plot



Şekil 12.65



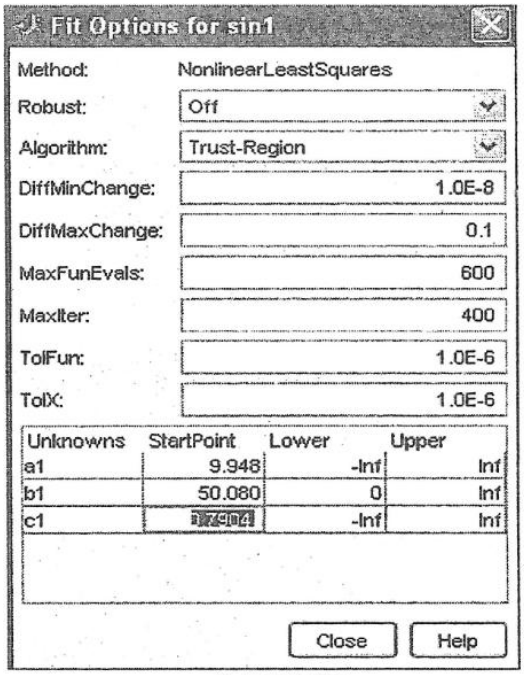
Şekil 12.66

seçenekleri kullanıldığında ise şekil 12.68 elde edilir. Yeni koşullarda Fitting penceresine ilişkin yeni değerler ise şekil 12.69'de görülmektedir.

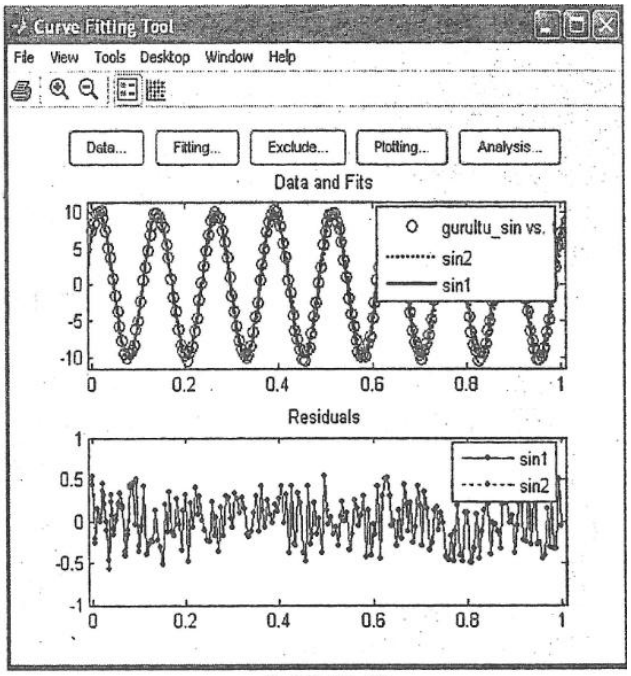
Şekil 12.60 ile şekil 12.69'daki istatistiksel değerler (SSE, R-square) değerleri karşılaştırıldığında önemli bir iyileşme sağlandığı görülecektir. Yeni çalışma durumunda, şekil 12.65'deki al=9.948, a2=50.08 ve a3=0.7904 katsayılarının, şekil 12.69'da; al=9.996, a2=50.28 ve a3=0.7744 değerlerine dönüştüğü de gözden kaçmamalıdır

Açıklama: Verilen bir data dosyası içinde yer alan vektörler içinde Inf veya NaN özellikli sayılar var ise bu sayılan vektörler içinden uzaklaştırmak için aşağıdaki komut satırları kullanılabilir. Örneğin a adlı bir vektör içinde hem Inf hem de NaN özellikli sayılar olsun. Bu vektör içindeki Inf sayıları tespit etmek için isinf komutu, Nan sayıları tespit etmek için ise isnan komutu kullanılır. Bu sayıların bulunduğu sütun numaraları f ind komutu ile tespit edildikten sonra bu sayılar boş kümeye atanırlar. Aşağıdaki komut satırları incelenmelidir.

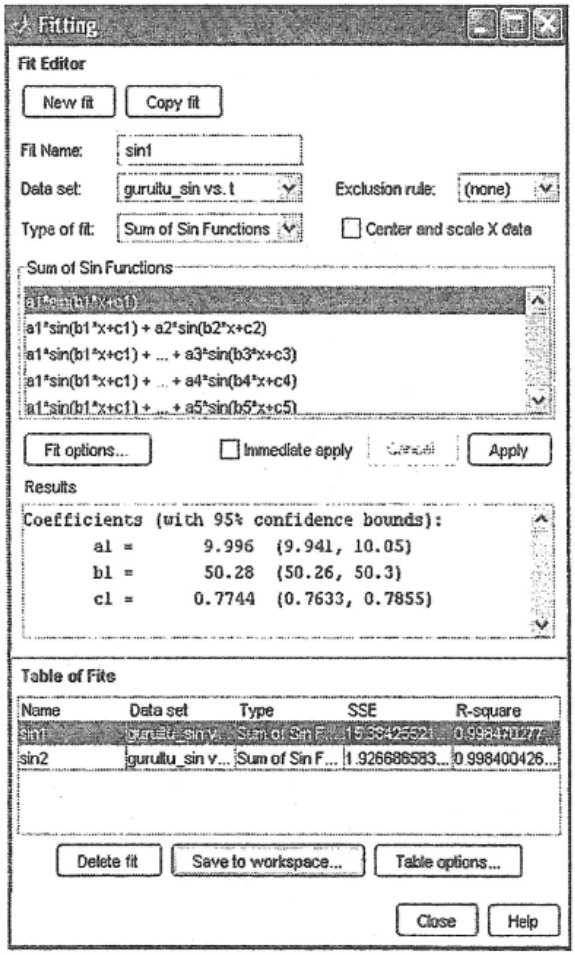
```
»a-[-l 3i -56 inf 0 pi NaN 34];          ('enter')
»indisl=find(isinf (a)) ;                ('enter')
»a (indis 1) = [ ] ;                     ('enter')
»indis2=f ind {isnan (a) ) ;             ('enter')
»a(indis2) = [ ];                        ('enter')
»a
   -1.    0 + 3.i    -56.    0    3.1416    34.
```



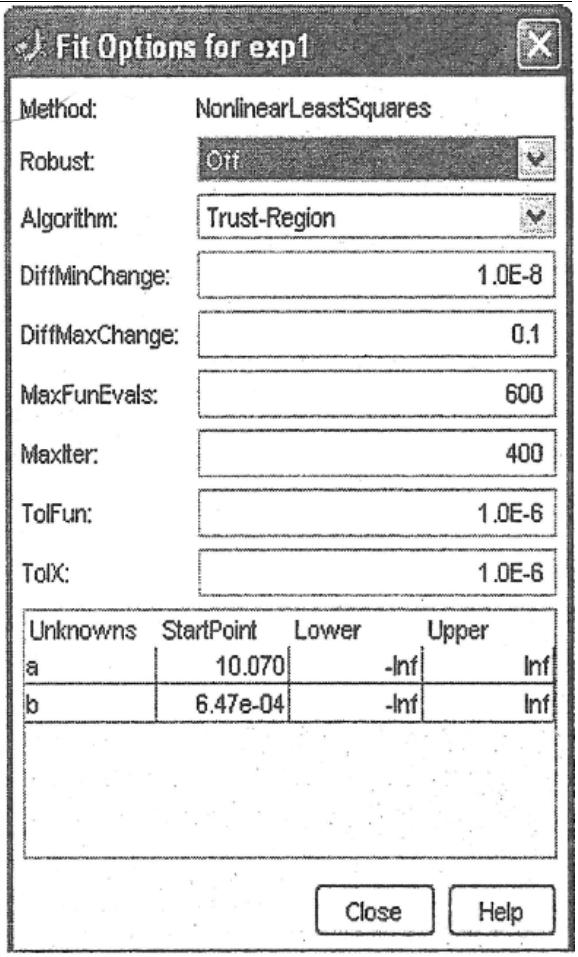
Şekil 12.67



Şekil 12.68



Sekil 12.69



Sekil 12.70

12.7.1.2.8.2. Fit Options penceresi

Fit Options penceresi, eğri uydurmada işlemindeki yöntem ve parametrelerin tespit edilmesinde kullanılır. Şekil 12.70'de görülen Fit Options penceresi içinde kullanılan seçenekler, şekil 12.69'da Type of fit içinde yer alan fonksiyon ve seçeneklere bağlı olarak değişir. Örneğin, Type of fit içinde yer alan Interpolant ve Smooting spline seçeneklerinde, Fit Options seçenekleri ortadan kalkar. Şekil 12.70'de görülen Fit Options penceresi, Type of fit olarak Exponantial seçeneğine göre oluşturulmuştur. Bu pencerede görülen terimler aşağıda açıklanmıştır:

- **Method**
  Eğri uydurmada kullanılan yöntemdir. Type of fit içinde seçilen eğri modeline göre, method otomatik olarak belirlenir. Örneğin lineer modeller için LinearLeas t Squares metod, nonlineer sistemler için Noml inearLeas t Squares metodu kullanılır.

- **Robust**
  Robust least squares eğri uydurma metodunun tercih edilip edilmeyeceğine karar vermek için kullanılır.
  Bu seçenek içinde 4 alt seçenek yer alır.
  -Off
  Bu seçenek default (kendiliğinden ayarlı) dır. Bu seçenek işaretli olduğunda Robust fitting işlemi yapılmaz.
  -On
  Robust methot (bisquare weights) kullanılır.
  -LAR
  Least **A**bsolute **R**esiduals ile minimizasyon işlemi yapılarak eğri uydurulur. -Bisquare
  Minimizasyon işlemi, summed square of the residuals (bisquare weighting) yöntemi ile yapılır. Çoğu durumda robust, eğri uydurmada en iyi seçenektir.

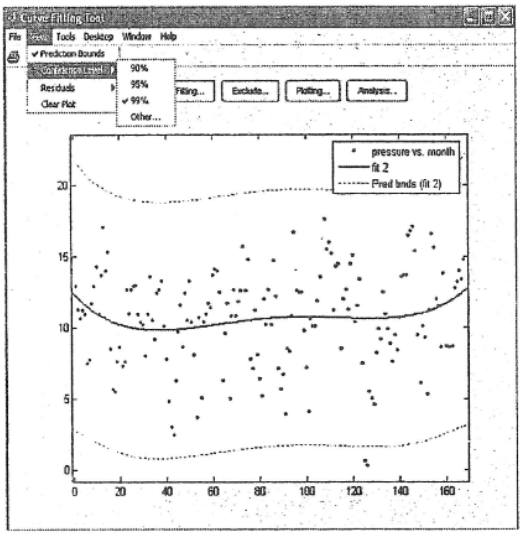
- **Algorithm**
  Eğri uydurmada kullanılacak algoritma seçeneklerini sunar. 3 adet algoritma içerir:
  -Trust-Region
  -Levenberg-Marquardt
  -Gaus s-Newt on

Şekil 12.70'de yer alan Dif fMinChange ve Dif fMaxChange seçenekleri, sonlu fark parametrelerini içerir. Dif fMinChange; sonlu fark Jacobianlar için katsayılardaki minimum değişim miktarı (default değeri 10^-8 dir), Dif fMaxChange ise sonlu fark Jacobianlar için katsayılardaki maksimum değişim miktarıdır (default 0.1 dir). Şekil 12.70'de yer alan MaxfunEvals, MaxIter,TolFun ve TolX seçenekleri ise eğri uydurmada kullanılan yakınsama kriterlerini içerir. MaxfunEvals; maksimum sayıda model fonksiyon sayısıdır. Default değeri 600 dür. Maxi t er; eğri uydurmada kullanılacak maksimum iterasyon sayısıdır. Default değeri 400 dür. TolFun (default değeri: 10^{A}-6) ve TolX (default: 10^{A}-6) ise sırası ile fonksiyon ve katsayılar için bitirme toleransıdır.Şekil 12.70'de yer alan (en alt pencere) seçenekler katsayı

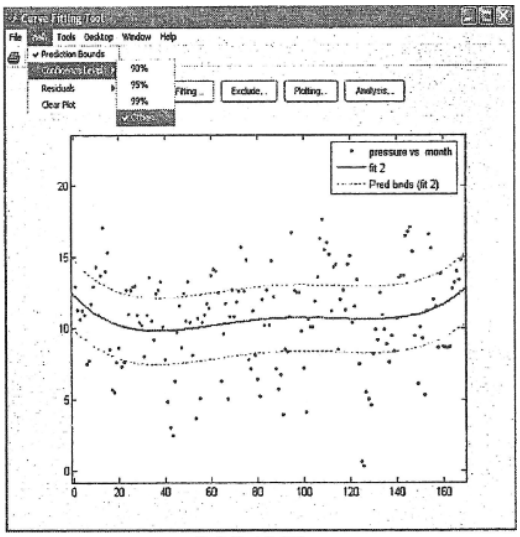
parametrelerine ilişkindir. Unknowns; uydurulan eğriye ilişkin katsayılardır. StartPoint; katsayı başlangıç değerleridir. Default değerleri seçilen modele bağlıdır.Lower; uydurulan eğri katsayılarının alt sınırıdır, Gaussians için 0'dan küçük olamaz. Upper; uydurulan eğri katsayılarının üst sınırıdır.

12.7.1.2.8.3. Prediction Bounds ve Confidence Level

Eğri uydurmada bulunan denklem katsayıları (coefficients) belirli aralık ile birlikte verilir. Örneğin şekil 12.69'da al (=9.996) katsayısının 9.941 (min) ile 10.05 (max) arasında olduğu görülmektedir. Aynı pencerede confidence bounds değeri olarak %S5 seçildiği görülmektedir. Kullanıcı isterse confidence bounds değerini şekil 12.71'de gösterildiği gibi Curve Fitting Tool penceresini kullanarak da seçebilir. Şekil 12.72'de ise confidence bounds değeri olarak %50 seçilmiştir. Her iki şekilde de noktalı çizgiler, uydurulan eğrinin alt ve üst sınırlarını göstermektedir. Her iki şekilde de görüldüğü gibi confidence bounds değeri azaldıkça noktalı çizgiler asıl eğriye (düz çizgili) yaklaşmaktadır, confidence bounds değeri büyüdükçe a1'in alt ve üst sınır değerleri arasındaki fark artacaktır, confidence bounds değerinin ne anlama geldiğini açıklamak için şöyle bir örnek verilebilir: Mevcut x ve y ekseni data değerlerini kullanarak bir eğri uydurulmuş olsun. Daha sonra (örneğin) x=5 yeni bir x değerine karşı gelen y değeri sorulsun. Bu işlemi yapabilmek için daha önce elde edilen (uydurulan) eğri denkleminden faydalanılacağı aşikardır. Eğer eğri denklemini elde etmeden önce confidence bounds değeri olarak (örneğin) %95 seçilmiş ise, x=5 değerine karşı gelen y değeri, %95 olasılıkla şekil 12.71'de gösterilen iki kesik eğri arasındaki bölge içinde kalacaktır. Eğer %50 seçilirse x=5 değerine karşı gelen y değeri %50 olasılıkla şekil 12.72'de gösterilen iki kesik eğri arasındaki bölge içinde kalacaktır. Görüldüğü gibi confidence bounds değeri büyüdükçe (y değerinin içinde kalacağı) bant aralığı genişlemektedir (aranan y değeri daha geniş sınırlar arasında aranmaya başlanmaktadır). Her iki şekilde de View menüsü içindeki prédiction bounds seçeneğinin işaretli olduğu unutulmamalıdır. Eğer bu seçenek işaretlenmez ise her iki şekilde yer alan kesik çizgiler ortadan kalkar. Şekil 12.69'da Resul t s penceresine bakarak, üstteki kesik çizgili eğrinin; y=9.941*x^{A}2+50.26* x+0.7633, alttaki kesik çizgili eğrinin ise y=10. 05*x^{A}2+50.3 *x+0.7855 denklemi ile gösterildiği söylenebilir. Düz çizgiye (asıl eğri) ilişkin denklem ise y=9.996*x^{A}2+50.28*x+0.7744 eşitliği ile verilmiştir.



Şekil 12.71



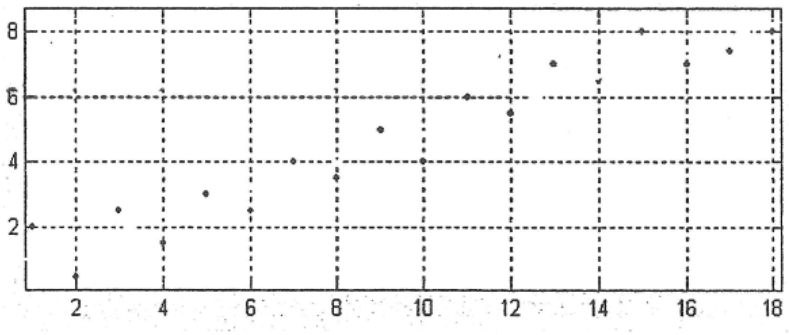
Şekil 12.72

12.7.1.2.8.4. Residual hakkında

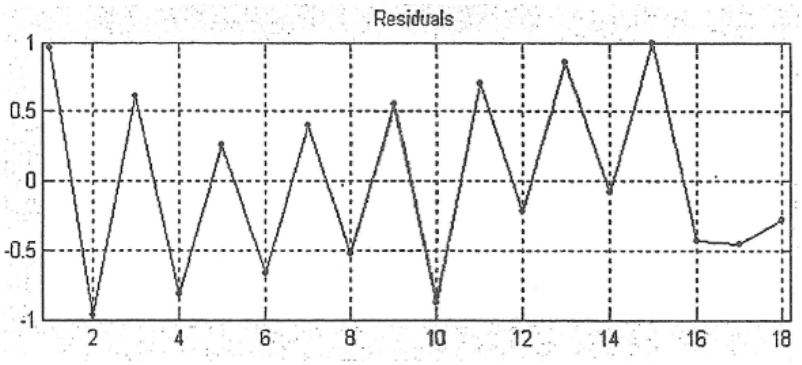
Residual (rezidü) vektörü, ölçüm sonucu bulunan y vektörü ile eğri uydurma sonunda elde edilen y vektörü arasındaki farka eşittir. Residual değişimim çizmek için Curve Fitting Tool penceresinde yer alan View/Residual menüsüne girilmelidir. Rezidü değişim eğrisi iki şekilde olabilir:

xxi Örneğin, data değerleri (ölçüm değerleri) şekil 12.73'de gösterildiği gibi değişiyor ise, bu değerlere ilişkin rezidü eğrisi şekil 12.74'de gösterildiği formda olacaktır (bu tip eğriler **iyi bir eğri uydurma** yapıldığına işaret eder). Burada görüldüğü gibi rezidü eğrisinin değişimi, y=Q eğrisinin etrafında dolaşmaktadır.

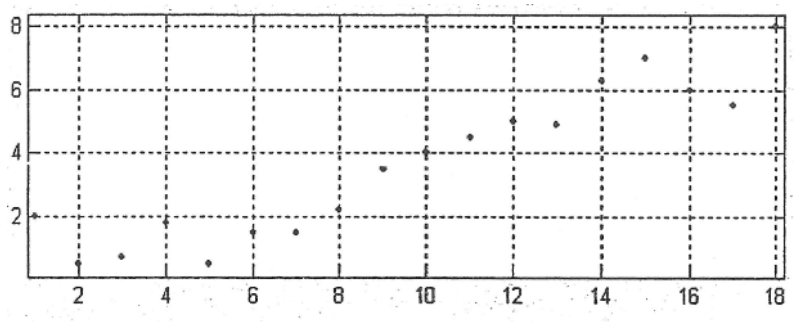
xxii Örneğin, data değerleri (ölçüm değerleri) şekil 12.75'de gösterildiği gibi değişiyor ise, bu değerlere ilişkin rezidü eğrisi şekil 12.76'da gösterildiği formda olacaktır (bu tip eğriler **kötü bir eğri uydurma** yapıldığına işaret eder). Burada görüldüğü gibi rezidü eğrisi (şekil 12.74'ün aksine), y=0 eğrisinin etrafında dolaşmamaktadır. Böyle bir rezidü eğrisi ile karşılaşıldığında yapılacak şey; eğri modelinin, **Fitting** penceresindeki Type of **Fit** alt penceresi içinde yer alan diğer eğri modellerinden birisi ile değiştirilmesidir.



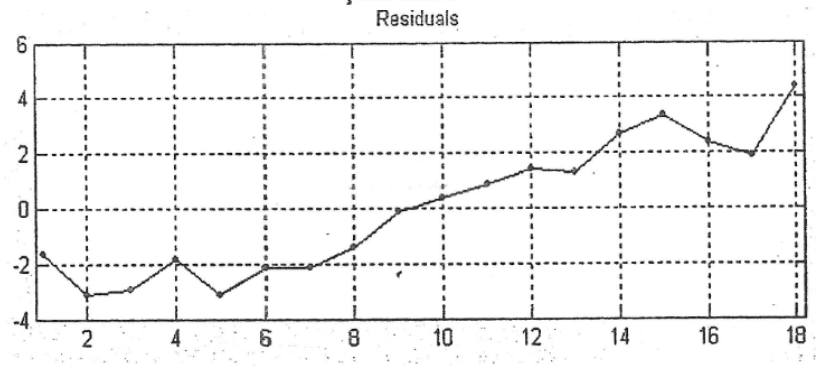
Şekil 12.73



Şekil 12.74



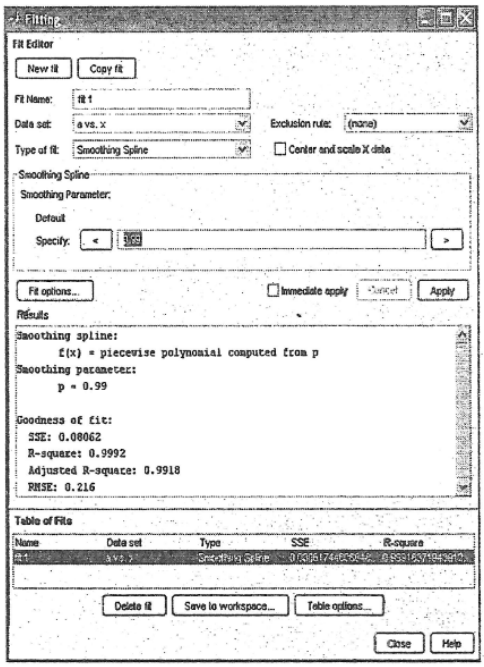
Şekil 12.75



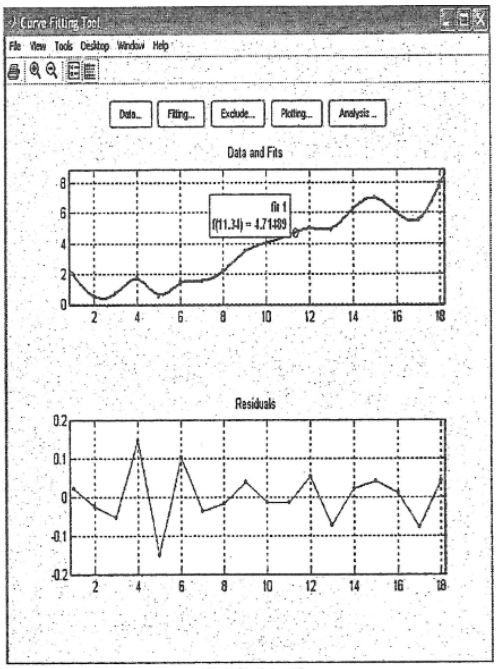
Şekil 12.76

12.7.1.2.8.5. Parametrik olmayan eğri uydurma

Şimdiye kadar parametrik eğri uydurma yaklaşımları tanıtıldı. Bazen kullanıcı ölçüm sonucu elde edilen data değerleri ile uydurulan eğri arasındaki uyumun çok daha düzgün olmasını arzu edebilir. Böyle bir durumda cftool ortamı kullanıcıya Interpolant ve Smooting Spline seçeneklerini sunar. Bu iki seçenek Fitting penceresindeki Type of Fit alt penceresi içinde yer alır. Bu iki seçenek kullanıldığında şekil 12.77'de görüldüğü gibi, eğri denklemi (katsayılan) olmaz (Fitting penceresinin Result alt penceresinde uydurulan eğriye ilişkin yalnız, istatistiksel katsayı değerleri bulunur). Kullanıcı bu iki modeli kullanarak eğriye ulaştığında, eğri üzerine fare ile gelip sol tuşa basarsa, açılan küçük pencere içinde bu noktanın x ve y değerlerini görür (şekil 12.78). Smooting Spline modelinde düzgün bir eğri uydurmak için Smooting parameter (p) seçeneği kullanılır (örneğin, şekil 12.77'de p=0.99 alınmıştır). Interpolant seçeneği altında kullanıcıya 4 farklı alt model sunulur: linear, nearest neighbor, cubic spline ve shape-preserving.



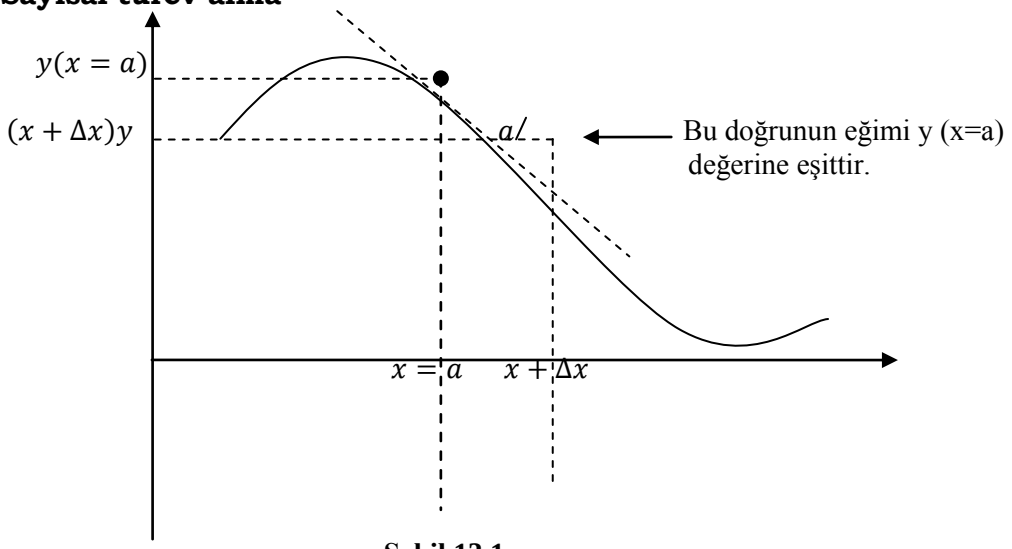
Şekil 12.77



Şekil 12.78

BÖLÜM 13

13.2. Sayısal türev alma



Şekil 13.1

y(x) fonksiyonunun x'e göre türevi, y'deki değişimin x'deki değişime oranı anlamına gelir. y'(x) ile gösterilir.

$$y'(x) = \frac{dy(x)}{dx} \qquad (13.1)$$

y'(x) değeri, y(x) fonksiyonuna x noktasında çizilen teğetin eğimi olarak ifade edilir. Şekil 13.1'de x=a noktasındaki y(x=a) fonksiyonunun türevi ; y'(x=a) ile gösterilmektedir. y(x) fonksiyonunun x noktasındaki türevi matematiksel olarak;

$$tga = y'(x) = \lim_{\Delta x \to 0}\left(\frac{y(x)-y(x+\Delta x)}{x-(x+\Delta x)}\right) = \lim_{\Delta x \to 0}\left(\frac{y(x)-y(x+\Delta x}{-\Delta x}\right) \qquad (13.2)$$

eşitliği kullanılarak hesaplanır. Sayısal türev işlemlerinde x_k noktasında çizilen teğetin eğitimi üç farklı noktada alınabilir.

- Şekil 13.2(a)'da x_{k-1}($x_{k'nın}$ sol tarafındaki nokta) ve x_k noktaları gözetilerek eğim hesabı yapılmaktadır (geri fark yaklaşımı). Şekil 13.2 (a)'dan ;

  $$tg(a) = y'(x_k) = \frac{y(x_k)-y(x_k-1)}{x_k-x_{k-1}} \qquad (13.3)$$

  elde edilir. (13.3) eşitliğinde $\Delta x = x_k - x_{k-1}$ olarak alınırsa;

  $$y'(x_k) = \frac{y(x_k)-y(x_k-1)}{\Delta x_k} \qquad (13.4)$$

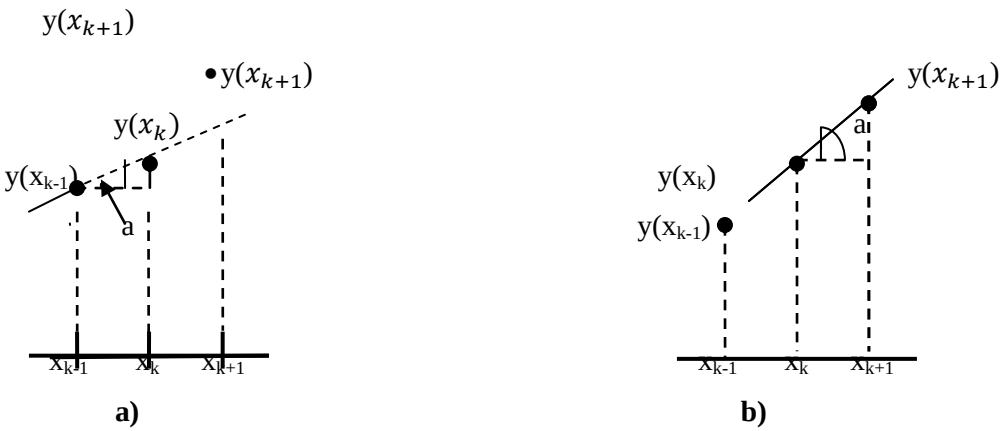
  elde edilir.

- Eğim hesabı x_{k-1} ve x_k noktaları arasında yapılmaz, şekil 13.2(b)' de gösterildiği gibi x_k ve x_{k+1} noktaları arasında yapılırsa (**ileri fark yaklaşımı**);

  $$tg(a) = y'(x_k) = \frac{y(x_{k+1})-y(x_k)}{x_{k+1}-x_k} \qquad (13.5)$$

  elde edilir. (13.7) eşitliğinde $\Delta x = x_{k+1} - x_k$ olarak alınırsa;

  $$y'(x_k) = \frac{y(x_{k+1})-y(x_k)}{\Delta x} \qquad (13.6)$$



Şekil 13.2

elde edilir. (13.4) ve (13.6) eşitliklerinin her ikisi de y(x) fonksiyonunun x_k noktasındaki türevi olmasına rağmen farklı sonuçlar elde edilebilmektedir. Eğer Δx mesafesi azaltılır ise (13.4) ve (13.5) eşitlikleri arasındaki fark azalırken, Δx mesafesi artırılır ise bu iki eşitlik arasındaki fark da artmaktadır.

- (13.4) ve (13.6) eşitlikleri yerine (**merkez fark yaklaşımı**);

  $$y'(x_k)=\frac{1}{2}\left(\frac{y(x_{k+1})-y(x_k)}{\Delta x} + \frac{y(x_k)-y(x_{k-1})}{\Delta x}\right) = \frac{y(x_{k+1})-y(x_{k-1})}{2\Delta x} \qquad (13.7)$$

  eşitliği kullanılarak, y(x) fonksiyonunun $x=x_k$ noktasındaki türevinin hesaplanmasına ilişikin hesaplama hatası minimize edilebilir. (13.7) eşitliğinin payında görüldüğü gibi birbirini takip eden iki sayı arasındaki fark değil , x_{k+1} ile x_{k-1} arasındaki fark hesaplanmaktadır.

MATLAB ortamında sayısal türev işlemi için (yukarıda verilen üç yöntemden ilk ikisi için) diff komutundan faydalanılır. diff komutu, bir vektörün bitişik iki değer arasındaki farkını hesaplayarak tüm vektörü tarar ve x vektörünün boyutundan bir eksik boyutta yeni bir vektör bulur.

diff(x): x vektörünün (tüm elemanları için) bitişik iki değeri arasındaki farkı kullanarak Δx fark vektörünü hesaplar. Fark vektörü; Δx=[x(2)-x(1) x(3)-x(2)... x(n)-x(n-1)]farklarında oluşur. Bu vektörde kullanılan n değeri length(x) olup, x vekötörünün eleman sayısını gösterir.

Aşağıdaki örnek incelenmelidir:

```
>> x=[-2,0,3,7,11,16];    %boyut 6
>> y=[1,3,8,12,18,25];    %boyut 6
>>   dx=diff(x) %Δx fark vektoru hesaplanıyor
dx =
    2     3     4     4     5     %boyut 5
>>  dy=diff(y)  %Δy fark vektoru hesaplanıyor
dy =
    2     5     4     6     7     %boyut 5
```

y'(x) değeri; y(x)' deki değişimin x'deki değişime oranı olduğuna göre yukarıda hesaplanan 'dy' değerinin 'dx' değerine oranı y'(x) türevine eşit olacaktır;

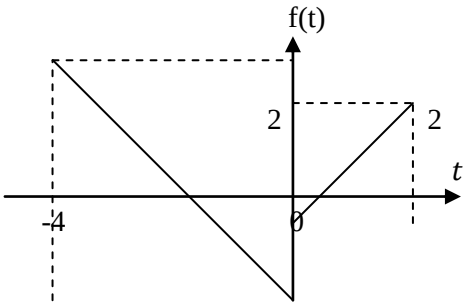
```
>> df =diff(y)./diff(x)     %türev hesaplanıyor
df =
   13.0000    13.6667    13.0000    13.5000    13.4000
>> xd =x(2:end)

xd =
    0     3     7    11    16
```

olur. x vektörü içinde yer alan ilk eleman değeri '-2' noktasındaki k=1 için x=-2 olmaktadır. k-1=0 için ise (x vektörü içinde sıfırıncı eleman olmayacağından) x vektöründe bu eleman tanımlı değildir. Dolayısı ile k=2 için x=0 değeri ile hesaplanmalara başlanmalıdır. Bu nedenle yukarıdaki program satırında x(2:end) ifadesi kullanılmaktadır. Yukarıdaki yaklaşım, x vektörünün ilk elemanı olan (-2) kesilerek x vektörü oluşturduğu için **geri fark yaklaşımı** olarak adlandırılır.

Aşağıdaki yaklaşım ise x vektörünün son elemanı (5) kesilerek x vektörü oluşturduğu için **ileri fark yaklaşımı** olarak adlandırılır:

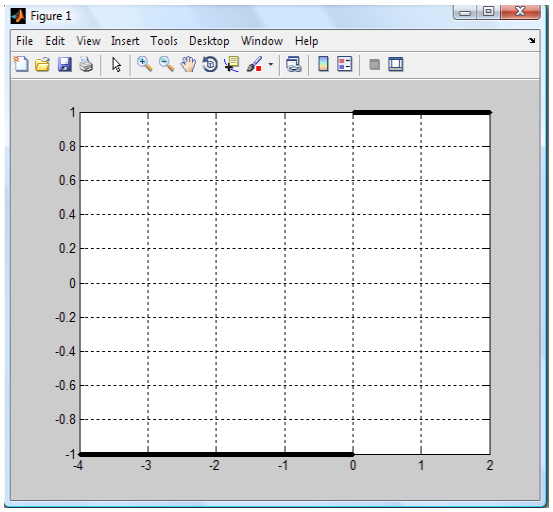
```
>> xd=x(1:end-1)
xd =
   -2     0     3     7    11
```

k=5 için x(5)=11 olduğu için k+1 için x vektöründe bir eleman tanımlı olmayacağından, en fazla k=4 için işlem yapılabilir. Bu nedenle yukarıdaki program satırında x(1:end-1) ifadesi kullanılmıştır.



Şekil 13.3

4



Şekil 13. 4

Problem 13.1

Şekil 13.3'de verilen f(t) eğrisinin (df(t)/dt) türevini geri fark yaklaşımı ile hesaplayarak çizdiriniz.

Çözüm

```
t=-4:0.01:2; k=length(t);
for m=1:k
    if t(m)>=-4 & t(m)<=0
        y(m)=-t(m);
    else
        y(m)=t(m);
    end
end
dt=diff(t);dy=diff(y);turev_y_t=diff(y)./diff(t);
plot(t(2:k),turev_y_t,'k.'),grid
```

yukarıda verilen programın çalıştırılması sonucunda elde edilen eğri şekil 13.4'de verilmiştir.

Problem 13.2

Bir cismin hız değişimi, $v(t)=3t^2+4t+6$ m/s olarak verilmiştir. Hareketlinin t=3. saniyedeki **a)** ivmesini dv/dt formülü ile hesaplayınız. **b)** a=diff(v)./diff(t) MATLAB komutu ile hesaplayınız.

Çözüm

a) $a=\frac{dv}{dt}=\frac{d(3t^2+4t+6)}{dt}=6t+4|\sum_{t=3}=22\text{m/sn}^2$

bulunur.

b) >> t=0:0.1:3; %değişken

```
>> v=3*t.^2+4*t+6;          %türevi alınacak fonksiyon
>> dt=diff(t); dy=diff(v);
>> a=diff(v)./diff(t);      %tüm zaman aralığı boyunca ivme değerleri bulunuyor
>> ivme_3saniye=a(end)      %3.saniyedeki ivme hesaplanıyor
ivme_3saniye =
   213.7000
```

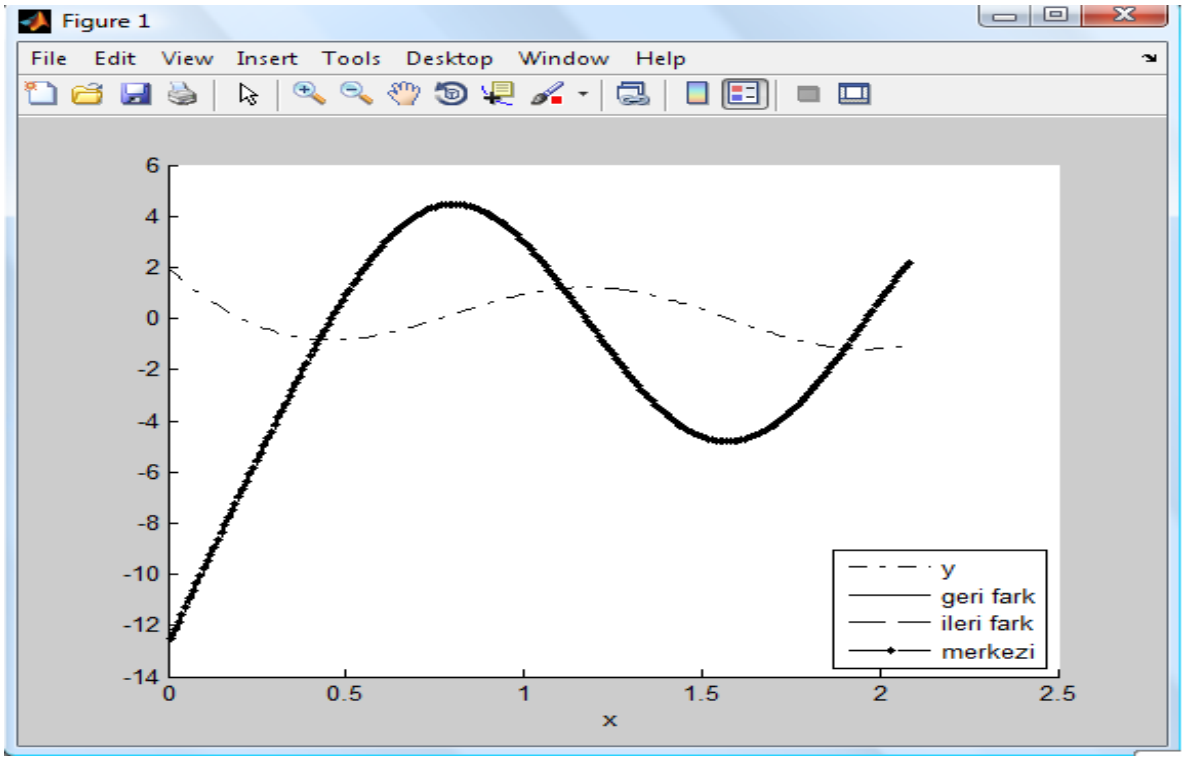
Bulunan ivme değeri, gerçek ivme ($22 m/sn^2$) değerine yakındır. Eğer yukarıda verilen programda t=0:0.01:3 kullanılırsa a=213.970 m/sn^2 elde edilir. Fakat t=0:0.0001:3 kullanılırsa a=213.9997 m/sn^2 elde edilir. Bu sonuca bakarak, diff komutu ile elde edilen türev değerlerinin, değişkenin (t) **artış aralığı** ile çok yakından alakalı olduğu söylenebilir.

Problem 13.3

y=$2e^{-4x}$-13.2*sin(4x) denkleminin x:0: 2*pi/3 aralığında y(x) eğrisini, türevini geri fark, ileri fark ve merkezi fark yaklaşımları ile çizdiriniz.

Çözüm

Program satırlarında görüldüğü gibi merkezi farkla türevde *diff* komutu kullanılmaktadır. Bunun nedeni merkezi farkın iki farkla çalışmasına rağmen *diff* komutunun bir farkla çalışmasıdır. y(**3:n**) ile y(**1:n-2**) elemanları arasında iki tane eleman bulunmaktadır. Şekil 13.5'de verilen programın sonunda elde edilen çizim penceresi görülmektedir.



Şekil 13.5

```
x=0:0.01:2*pi/3;
n=length(x);
y=[2*exp(-4*x)-13.2*sin(4*x)]; hold on
 plot(x,y,'k-.')        %y(x)eğrisi çizdiriliyor
turev=diff(y)./diff(x);          %ileri veya geri fark için türev alınıyor
plot(x(2:end),turev,'k-')      %geri fark türev eğrisi
plot(x(1:end-1),turev,'k--')            %ileri fark türev eğrisi
merkezi=(y(3:n)-y(1:n-2))./(x(3:n)-x(1:n-2));        %merkezi türev ifadesi
plot(x(2:n-1),merkezi,'k.--')            %merkezi fark türev eğrisi
legend('y','geri fark','ileri fark','merkezi',4),
xlabel('x')
```

Problem 13.4

Problem 13.3'de verilen y(x) eğrisinin geri fark yaklaşımı altında türevinin kritik noktalraını bulunuz.

Çözüm

```
>> x=0:0.01:2*pi/3;
>> y=[2*exp(-4*x)-13.2*sin(4*x)];
>> turev=diff(y)./diff(x);
>> xd=x(2:length(x));        %boyutu(n-1) dir (geri fark alınıyor)
>> carpim=turev(1:length(turev)-1).*turev(2:length(turev));
>> kritik=xd(find(carpim<0))        %iki eğrinin çarpımının işaret değiştirdiği yerler bulunuyor
kritik =
   0.4600    13.1700    13.9600
```

Kritik noktalar elde edilirken, türev eğrisi biraz sağa cu ile elde edilen eğri ile türevin biraz sola kaydırılması ile elde edilen eğri birbirleri ile çarpılmaktadır. Bu çarpımda sonucun pozitiften negatife yada negatiften pozitife geçtiği noktalar kritik noktalardır. Her iki geçiş durumunda *carpım<0* olmalıdır. Yukarıda carpım ve kritik satırları bahsedilen işlemlerin yapıldığı MATLAB satırlarıdır. Elde edilen kritik değerler ile y(x) eğrisi karşılaştırıldığında bu değerlerin y(x) eğrisinin kıvrıldığı yerler olduğu görülecektir.

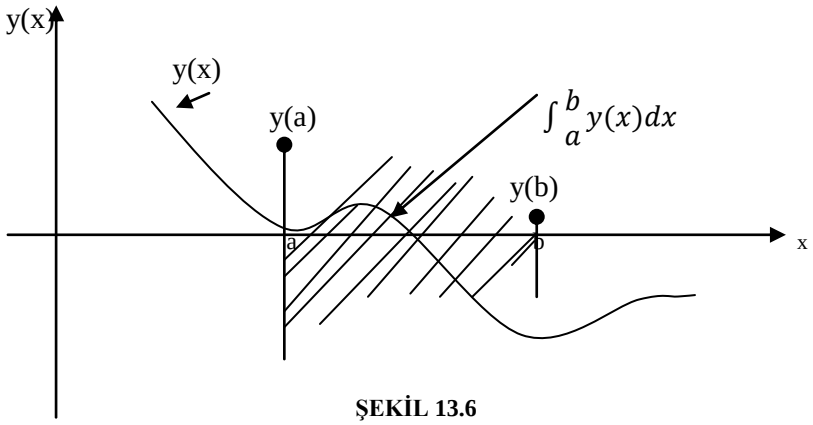
13.2 Sayısal entegrasyon

y(x) fonksiyonunun a ≤ x ≤ b aralığında **entegre** edilmesi, bu eğri ile x=a noktası ile x=b noktaları arasında kalan bölgenin **alanının hesaplanması** olarak değerlendirilir;

$$s = \int_a^b y(x)dx$$

(13.8)

Şekil 13.6'da y(x) eğrisi ve bu eğrinin entegrali gösterilmiştir. Yukarıda verilen entegralin değeri S olarak hesaplanmaktadır. Çoğu durumda bu entegralin değeri analitik olarak hesaplanabilirse de fazı y(x) fonksiyonlarının analitik olarak entegrali mümkün olmayabilir. Bu durumda sayısal entegral alma teknikleri kullanılarak y(x) eğrisinin entegrali alınabilir.



ŞEKİL 13.6

Sayısal entegral alma tekniklerinde entegral aralığı parçalara bölünerek her bir aralıkta fonksiyon yerine sabit 'doğrular' (yamuk şeklinde) yada 'paraboller' alınarak yaklaşık hesaplama yoluna gidilir.

13.2.1 Yamuklar yöntemi ile entegrasyon

Bu yaklaşımda [a,b] aralığı 'n' adet eşit paraya bölünür. Şekil 13.7 'de bu yaklaşım gösterilmiştir.[a,b] aralığı n parçaya bölünür ise her bir parçanın genişliği;

$$\Delta x = \frac{b-a}{n} \quad (13.9)$$

olur. Şekil 13.7'de ABCD arasında kalan yamuğun alanı;

$$S_i \approx (X_{i+1}-X_i)\frac{y(x_i)+y(x_{i_{+1}})}{2} = \frac{\Delta x}{2}(y(x_i)+y(x_{i_{+1}})) \quad (13.10)$$

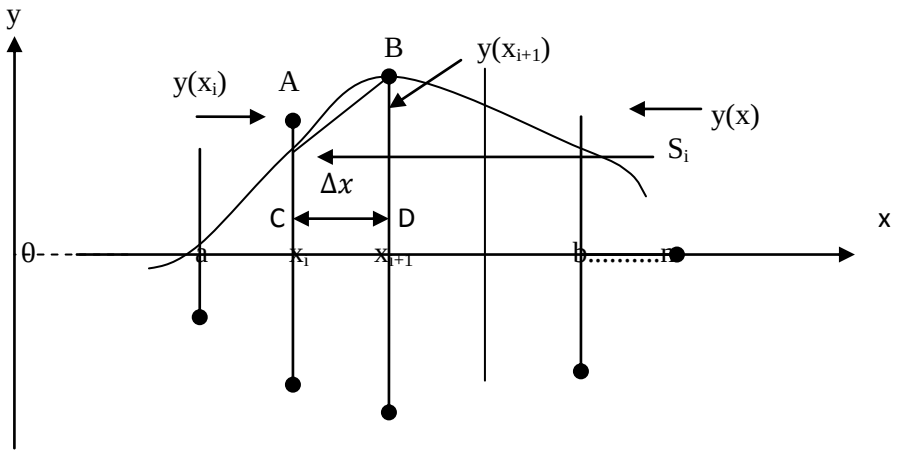
olur. S_i alanı,

$$S_{igercek} = \int_{x_i}^{x_{i_{+1}}} y(x)dx \quad (13.11)$$

eşitliği ile hesaplanan gerçek alandan daha küçüktür. Şekil 13.7'de verilen y(x) eğrisinin <u>yamuklar yöntemi</u> yardımı ile hesaplanan tüm (yaklaşık) alanı;

$$S_{Top} = \sum_{i=0}^{n} \frac{\Delta x}{2}(y(x_i)+y(x_{i_{+1}})) = \frac{\Delta x}{2}(y(x_0)+2y((x_1)+..+2y((x_{n-1})+y(x_n)) \quad (13.12)$$

olarak bulunur. MATLAB ortamında bir eğrinin yamuklar yöntemi ile entegrali *trapz* komutu ile hesaplanır.



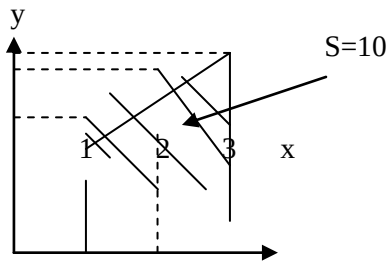
Şekil 13.7

S = trapz(x,y) : y(x) değişiminin entegralini yamuklar yöntemi ile hesaplar. x ve y aynı boyutta iki vektördür. Eğer x bir sütun vektörü ise y ; lenght(x) boyutuna uygun bir dizi olmalıdır.

S = trapz(y) : y'nin entegralini yamuklar yöntemi ile hesaplar. y bir vektör ise x değeri **1**'den itibaren **1**'er olarak artar. S bu yamuğun altındaki alanı hesaplar.

```
>> y=[3 5 7];
>> S=trapz(y)
S = 10
```

Yukarıdaki MATLAB satırlarının bulunduğu alan değeri şekil 13.8'de gösterilmiştir.



Şekil 13.8

Eğer S = trapz (y) komutundaki y bir matris ise S bir satır vektörü olur. S'nin her bir elemanı y'nin her bir kolonunun yamuk yöntemi ile alanını hesaplar. Bu durumda x değeri ise 1'er artar. Aşağıdaki örnek incelenmelidir;

```
>> y=[3 6 4; 5 10 12];
>> S = trapz(y)
S =
    4     8     8
```

Yukarıdaki MATLAB satırlarında x=1 için y'nin ilk sütunu; y=3 ve y=5 arasında kalan yamuğun alanı hesaplamakta ve 4 elde edilmektedir. Bu değer S'nin ilk elememanı olmaktadır. x=2 için y'nin ikinci sütunu; y=6 ve y=10 arasında kalan yamuğun alanı hesaplanmakta ve 8 elde edilmektedir. Bu değer S'nin ikince elemanı olmaktadır. Son olarak ise x=3 için y'nin üçüncü sütunu; y=4 ve y=12 arasında kalan yamuğun alanı hesaplanmakta ve 8 elde edilmektedir. Bu değer S'nin üçüncü elemanı olmaktadır.

S = trapz(y, 'dim') :	y bir matris ve dim=1 ise x'i 1'den başlayarak ve 1'er artırarak y'nin her bir sütunu için alanı hesaplar ve S vektörünün elemanı olarak atar. Bu işlemi y'nin tüm sütunları bitinceye kadar yapar. Eğer *dim=2* ise aynı işlemi y'nin satrıları için gerçekleştirir. Burada da x yine 1'er artırılır.

Aşağıdaki iki farklı örnek incelenmelidir:

```
>> y=[3 6 4; 5 10 12];
>> S = trapz(y)
S =
    4     8     8
>> y=[3 4 5;6 9 12];
>> S= trapz(y,1)
S =
   4.5000    6.5000    8.5000
>> S= trapz(y,2)

S =
    8
   18
```

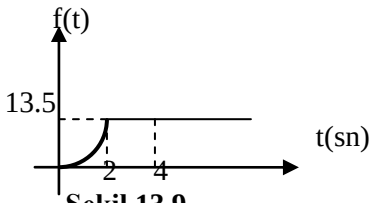
Aşağıda verilen program satırlarında, cos(x) eğrisinin x=-pi/2:pi/2 arasında kalan **alan** hesabı, bu aralık 10 eşit parçaya bölünerek ve yamuklar yöntemi kullanılarak yapılmaktadır:

```
>> x=linspace(-pi/2,pi/2,10);
>> y=cos(x);
>> alan = trapz(x,y)
alan =
   13.9797
```

Eğer yukarıda verilen program satırlarında , -pi/2:pi/2 aralığı 100 eşit parçaya bölünseydi alan=13.9998 olacaktı. Sonuç olarak hesap edilen alanın doğru olması için entegral adım aralığının yeterli küçüklükte seçilmesi gerekmektedir. 1000 eşit parça için ise alan=2 olmaktadır.

Problem 13.5

Şekil 13.9'da f(t)=$e^{0.4581*t} - 1$ eğrisi ile t ekseni arasında kalan alanı, t=[0:4] aralığında *trapz* komutunu kullanarak bulan bir m dosyası yazınız.



Şekil 13.9

Çözüm

```
t=0:0.01:4;
for  k=1:length(t)
   if  t(k)>=0 & t(k)<=2
       f(k)=exp(0.4581*t(k))-1;
else
       f(k)=13.5;
   end
end
alan=trapz(t,f)
```

Yukarıda verilen programın çalıştırılması sonunda Command Window ortamında elde edilen sonuç aşağıda verilmiştir.

```
>>  alan =
4.2739
```

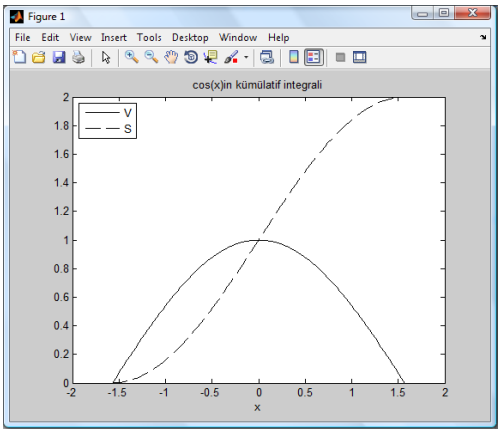
z=cumtrapz(x,y): Bu komut,y(x) eğrisinin yamuklar yöntemini kullanarak kümülatif entegral hesabını yapar. x ve y vektörleri aynı boyutta olamlıdır. Bu yaklaşımda x'in aldığı her değer için ayrı bir alan hesabı yapılır. Örneğin x=[1 3 7] ise bu komut sıra ile önce x=0 için alan hesabı yapar. Daha sonra x=1 için, y (x) eğrisinin 1'in sol tarafında kalan alanı hesaplar. Daha sonra x=3 için, y (x) eğrisinin 3'ün sol tarafında kalan alanı hesaplar.. Böylece bir sonraki x değerine ilişkin alan hesabında bir önceki x değerine ilişkin alan hesabı dahil olmaktadır. Aşağıdaki örnek incelenmelidir: *cumtrapz* komutunu, *trapz* komutundan farklı olarak adım adım integrasyon sonuçlarını vermesidir.

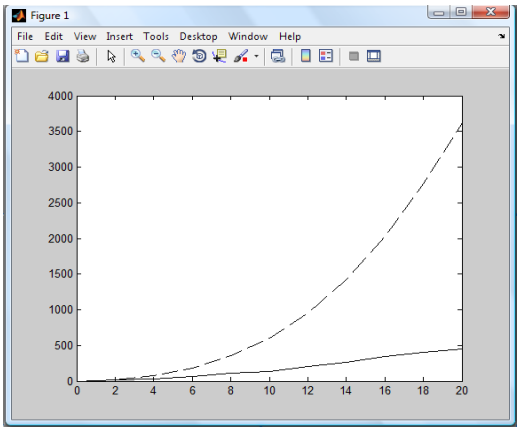
x = linspace(-pi/2,pi/2,100) ; y = cos(x); S = **cumtrapz**(x, y);

plot(x, y, ' k- ' , x, S ,' k--') , title('cos(x) in kümülatif integrali '),

xlabel(' x ') ; legend(' cos(x) ' , 'cos(x) in entegrali ', 2) , legend('V' ,'S' ,2)

Yukarıda verilen ve MATLAB eidtör ortamında yazılan programın uygulaması sonucu elde edilen grafik şekil 13.10'da gösterilmiştir.





Şekil 13.10 **Şekil 13.11**

Tablo 13.1

Zaman (s)	0	2	4	6	8	10	12	14	16	18	20
Hız(m/sn)	0	22	34	67	110	140	210	260	341	389	451

Yukarıdaki zaman-hız değerleri kullanarak hareketlinin zaman- yol-hı değerlerini bulan ve editör ortamında yazılan MATLAB programı aşağıda verilmiştir.

```
>> > t= [0:2:20];

V= [0, 22, 34, 67, 110, 140, 210, 260, 341, 398, 451];

S= cumtrapz(t, V);

disp ([' zaman Hiz Yol']);disp([t',V',S'])
```

plot (t, V, 'k-', t, S, 'k--')

zaman	Hiz	Yol
0	0	0
2	22	22
4	34	78
6	67	179
8	110	356
10	140	606
12	210	956
14	260	1426
16	341	2027
18	398	2766
20	451	3615

13.2.2 Parabolik (Simpson) yöntemi ile entegrasyon

Trapz komutu ile y(x) eğrisinin altında kalan alan 'yamuklar yöntemi' ile hesaplanmıştı. Burada ise [a,b] aralığı 2n adet eşit parçaya bölünecek ve üst kısmında 'doğru' şeklinde (yaklaşık bir) eğri yerine ikinci dereceden bir 'parabol' kullanılarak y(x) eğrisinin altında kalan alan hesaplanacaktır. Simpson yöntemi olarakda adlandırılan bu yaklaşımda alan değeri;

$S_t=\frac{\Delta x}{3}(y((x_0)+4y(x_1)+2y(x_2)+4y((x_3)+\cdots+2y(x_{2n-2})+4y(x_{2n-1})+y(x_{2n}))$ (13.13)

Eşitliği kullanılarak hesaplanır. Yukarıda x_k değeri alt entegrallerin son noktalarını göstermektedir. Bu ifade x_0=a, x_{2n}=b, $\Delta x=(b-a)/(2n)$ olmaktadır.

Parabolik bir eğri yerine daha yüksek bir eğri tercih edilirse bu yöntemin adı 'Adaptive Labatto' yaklaşımı olarak adlandırılır. Eğer entegre edilen fonksion bazı noktalarda tekillik (bu noktalarda eğrinin türevinde sonsuzluk ya da tanımsızlık) mevcut ise sayısal entegralden tatminkar bir sonuç elde edilmeyebilir.

MATLAB ortamında Simpson yöntemi için iki adet (kuadratik) komut kullanılır:

quad('fonksiyon', a,b): Simpson kuralı ile entegre edilecek eğri 'fonksiyon' ile gösterilen yere yazılır. Eğri a ve b arasında entegre edilir.Eğer giriş değerleri vektör olarak verilmiştir ise çıkış vektör olacaktır. Entegral sonucu inf ifadesini gösterir ise eğride 'tekillik' olduğu düşünülür.

quadl('fonksiyon', a,b): Adaptive Lobatto quadrature kuralı ile a-b aralığında entegre edilecek eğri fonksiyon ile gösterilen yere yazılır. Bu yaklaşım quad komutundan (tekillik problemlerini aşma açısından) daha elverişlidir.

aralıkta quad ve quadl komutları ile entegralleri aşağıda hesaplanmıştır:

```
>> quad ('cos(x)',-pi/2,pi/2)
ans =
```

```
   2.0000
>> quadl('cos(x)' , -pi/2,pi/2)
ans =
   2.0000
```

quadl (‘fonksiyon ‘ , a,b,’tolerans’), quad (‘ fonksiyon’ , a, b, ‘tolerans’) komutların da ise yukarıdaki açıklamalara ilaveten ‘ tolerans belirten sayı bulunmaktadır.Eğer böyle bir sayı komut içinde kullanılmamış ise tolerans olarak (default olarak) 1e-6 değeri alınır.

Örnek olarak $\int_2^8 \frac{1}{x^5+x^2+15x-6}$ dx entegrali sayısal integral komutları **üç** farklı şekilde hesaplanır.

1) Tek değişkenli bir fonksiyon direkt olarak ‘fonksiyon ’ yerine yazılır:

```
>> alan=quad('13./(x.^5+x.^2+15*x-6)',2,8);
```

2) İnline komutu ile:

```
>> F= inline ('13./(x.^5+x.^2+15*x-6)');
>> alan =quad(F,2,8)
```

3) fonksiyon’u alt programda tanımlayarak:

```
>> alan = quad(altfonk,2,8);
>> function y = altfonk(x)            %yukarıdaki satıra ilişkin alt program
>>   y=13./(x.^5+x.^2+15*x-6);
```

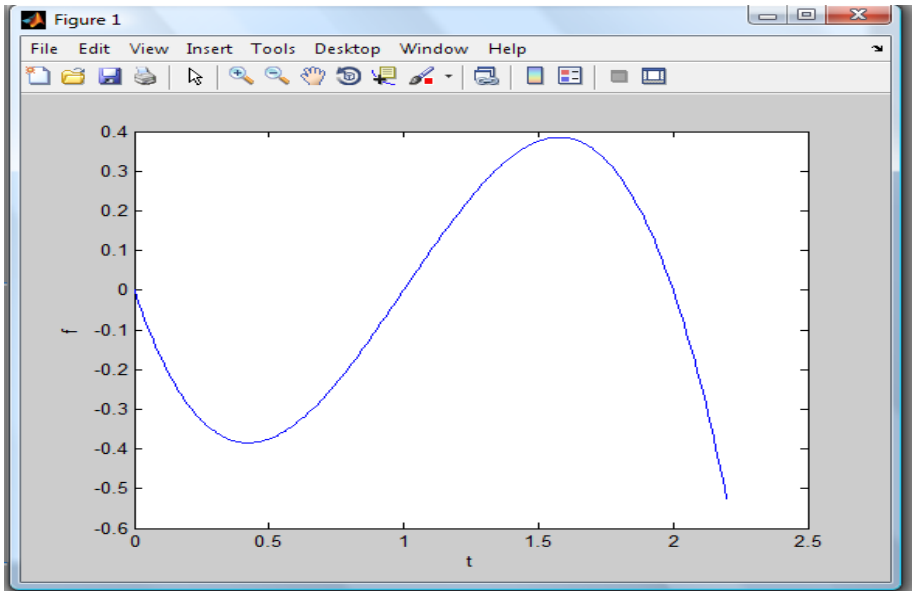
Problem 13.6

f(t) = $-t^3 + 3t^2 – 2t$ eğrisi ile t ekseni arasında t=0 : 2.2 saniye aralığında kalan **toplam alanı** hesaplayınız.

Çözüm

Öncelikle verilen eğri (t,f) düzleminde çizilmelidir:

```
>> t=0:0.01:2.2;
>> f = -(t.^3)+ 3*(t.^2)-(2*t);
>> plot (t,f),xlabel ('t'), ylabel ('f')
```



Şekil 13.12

Yukarıda verilen MATLAB programın çalıştırılması sonunda elde edilen grafik şekil 13.12'de gösterilmiştir. Şekilde 13.12 'de verilen eğrinin eğrinin Ot ekseni ile arasında iki adet alan bulunmaktadır. Bunlardan ilki [0 1] saniye aralığında diğeri ise [1 2] saniye aralığındadır. Eğer f(t) eğrisinin direkt olarak entegrali alınırsa iki alan değeri işaret olarak farklı olduğundan toplandıklarında problemde istenen toplam alan değeri bulunmuş olmaz ve sonuç **hatalıdır:**

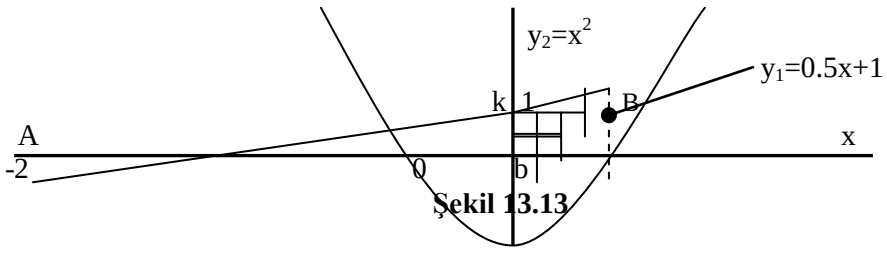
```
>> alan=quad('-(x.^3)+3*(x.^2)-(2*x)',0,2.2)
alan =
   -0.0484
```

Verilen eğri mutlak değeri alınarak entegre edilirse aranan toplam alan doğru bir şekilde hesaplnmış olur.

```
>> alan=quad ('abs (-(x.^3)+3*(x.^2)-(2*x))',0,2.2)
alan =
    0.5484
```

Problem 13. 7

Şekil 13.13'de verilen taralı alanı gösteren OKB alanını hesaplayan ve her iki eğriyi aynı eksen üzerine çizdiren MATLAB programını yazınız.(Not: Hiçbir hesaplama el ile yapılmayacak, hepsi MATLAB ortamında gerçekleştirilecektir.)



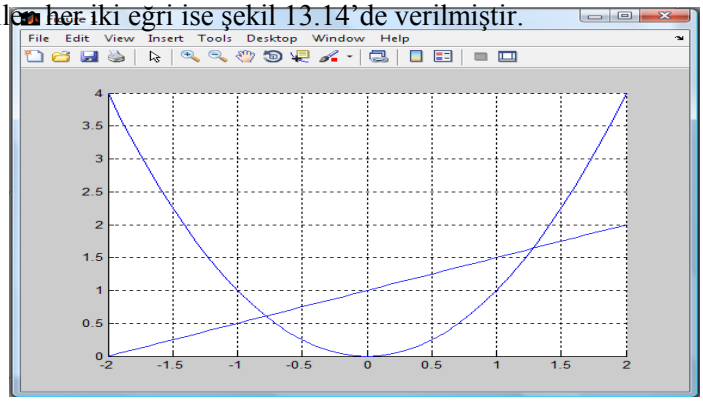
Şekil 13.13

Çözüm

```
a=roots([1 -0.5 -1])                % y1 ve y2 eğrilerinin kesiştiği noktalar aranıyor
b=a(2)                              % B noktasında apsis değeri
OKBb_yamugu=quad('0.5*x+1',0,b)
ObB_alti=quad('x.^2',0,b)           % yarım parabolün altındaki alan
alan =OKBb_yamugu - ObB_alti        % aranılan alan hesabı
x=-2:0.1:2,y=0.5*x+1;grid
hold on                             % iki eğri aynı eksen üzerinde çizdiriliyor
plot (x,y);y1=x.^2,plot(x,y1)
                                    % b değeri biliniyorsa,
```

Not: S=quad('0.5*x+1-x.^2',0,b)

Yukarıda verilen programın çalıştırılması sonunda elde edilen değerler aşağıda gösterilmiştir. Program sonunda elde edilen her iki eğri ise şekil 13.14'de verilmiştir.



Şekil 13.14

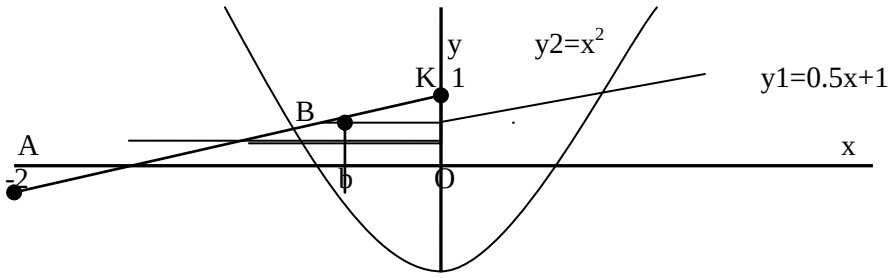
```
>>a =

    -0.7808
    13.2808
b =

    13.2808
OKBb_yamugu =

    13.6909
ObB_alti =
```

0.7003

alan =

0.9905

Problem 13.8



Şekil 13.15

Şekil 13.15'de verilen taralı alanı ABO alanının hesaplayan MATLAB programını yazınız. (Not: Hiçbir hesaplama el ile yapılacak, tüm hesaplamalar MATLAB ortamında gerçekleştirilecektir.)

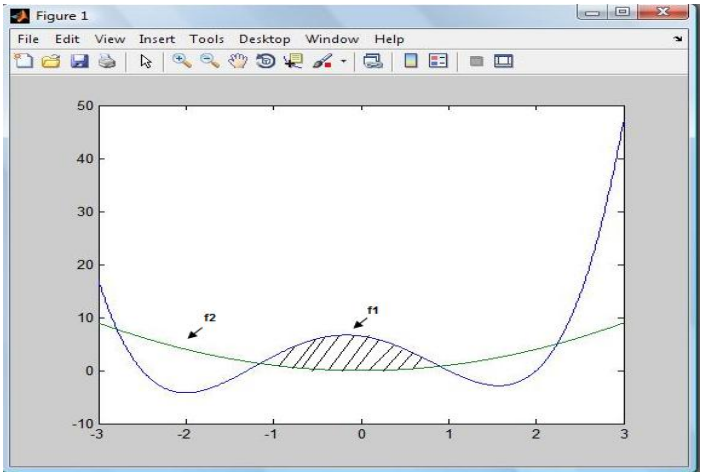
Çözüm

```
>> a=roots([1  0.5  -1])          % y1 ve y2 eğrilerinin kesiştiği noktalar aranıyor
>> b=a(1)                         % B noktasının apsis değeri
b=
        -0.78077640640442
>> BKO_alani=guad(' 0.5'*x+1-x.^2 ,b ,0)
>> AKO_alani=quad(' 0.5*x+1 ',-2 ,0)          %  üçgenin alanı
>> alan =AK0_alani-BK0_alani                  % aranılan alan hesabı

alan =
         0.5303
>> x=-5:0.1:5;y=0.5*x+1;
>> hold on;
>> plot(x,y);
>> y1=x.^2;
>> plot(x,y1)  % çizim sonucu verilmiştir.
```

Problem 13.9

Şekil 13.16'da **f1** eğrisi 4. Dereceden bir polinom olup katsayı vekötürü

[1 0.8 -6.15 -2.15 6.5] şeklindedir. **f2** eğrisi ise $y=x^2$ fonksiyonu ile verilmektedir. Her iki eğri [-3 :3] aralığında çizdiren ve şekil 13.16'da gösterilen taralı alanı hesaplayan programı yazınız.



Şekil 13.16

Çözüm

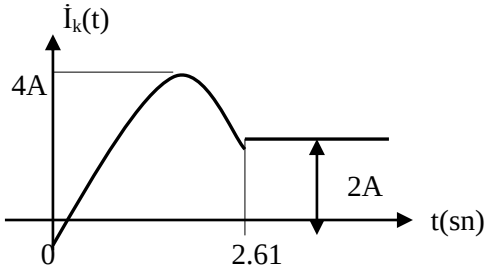
```
a=-3: 0.01: 3;
y=[1 0. 8  -6.15  -2.15  6.5 ];
f1=polyval(y,a);
f1=polyval(y,a);
f2=a.^2;
plot(a,f1,a,f2)              % her iki eğri çizdiriliyor (çizim sonucu verilmedi)
t=solve (' t ^4+0.8 *t ^3-6.15*t^2 -2.15*t+6.5-t^2=0 ' )  % f1-f2=0        (kök hesabı)
                             %kökler sembolik karakterden sayısal değere dönüştürülüyor
deger=double(t);             % entegral sınırları bulundu
alan=quad(' t. ^4+0.8*t. ^3-6.15*t. ^2-2.15*t+6.5-t. ^2',deger(2),deger(3))
                             % d(3) üst sınır,d(2) alt sınır
```

Yukarıda verilen program satırların çalıştırılması sonunda aşağıdaki değerler elde edilir.

```
>>
t =
         -2.78
         -13.15
           .9
         2.24
deger    =
         -2.78
         -13.15
         0.90
```

2.24

alan =
8.81



Şekil 13.17

Problem 13.10

Şekil 13.17'de verilen $i_k(t)$ değişiminin [0 5] sasniye aralığında (t=0:0.01:5 alınız);

a) Ortalama değerini bulan matlab programını yazınız.

b) Etkin değerini bulan MATLAB programını yazınız.

Çözüm

a)
```
t=0:0.01:5;
uzun = length(t); ik=zeros(1,uzun);
for k=1 :uzun
  if t(k)>=0 & t(k)<=2.61
     ik(k)=4*sin(t(k));
  else
     ik(k) =2;
  end
end
akim_ort=mean(ik);          % akım ortalamasını bulur
ort=trapz(t,ik)/5           % akım ortalamasını bulur(Alan/periyod=ortalama)
ort =
    2.4456
```

b) $\mathbf{I_{etkin}} = \sqrt{\frac{1}{5}\int_0^5 \boldsymbol{i}\frac{2}{\boldsymbol{k}}(\boldsymbol{t})}\mathrm{dt}$ *% etkin akım ifadesi*

13.2.3 İki boyutlu entegrasyon

S=dblquad (‘ fonksiyon ’ , xmin,xmax,ymin,ymax) :	fonksiyon yerine gelecek eğri denklemi iki değişkenli olan z=f(x,y) şeklinde olmalıdır. Dış entegralin alt sınır ymin,üst sınırı ymax alınmalıdır. İçteki entegralin alt sınırı xmin, üst sınırı xmax olmalıdır. dblquad komutunu kullanmak için öncelikle f(x,y) eğrisi tanımlanmalıdır.

Aşağıda verilen iki boyutlu entegral;

S=$\int_{ymin}^{ymax}\int_{xmin}^{xmax} f(x,y)dxdy$

(13.14)

MATLAB ortamında dblquad komutu hesaplanır.

Aşağıda command window ortamında , z=cos(2*x).*y.^3+1 fonksiyonu çizdirilmektedir. Program satırlarından elde edilen grafik şekil 13.18’de gösterilmiştir.

```
x=linspace( 0, 2, 15 )
y=linspace( -pi/3, pi/3,15 );
[xx,yy]=meshgrid (x,y);
zz=yuzey (xx,yy);                %yuzey adlı programa bakılsın
mesh (xx,yy,zz), xlabel (‘x’) , ylabel (‘y’),zlabel(‘z)
```

Aşağıda ise yuzey .m adlı MATLAB dosyasına yazılan altprogram satırları gösterilmiştir:

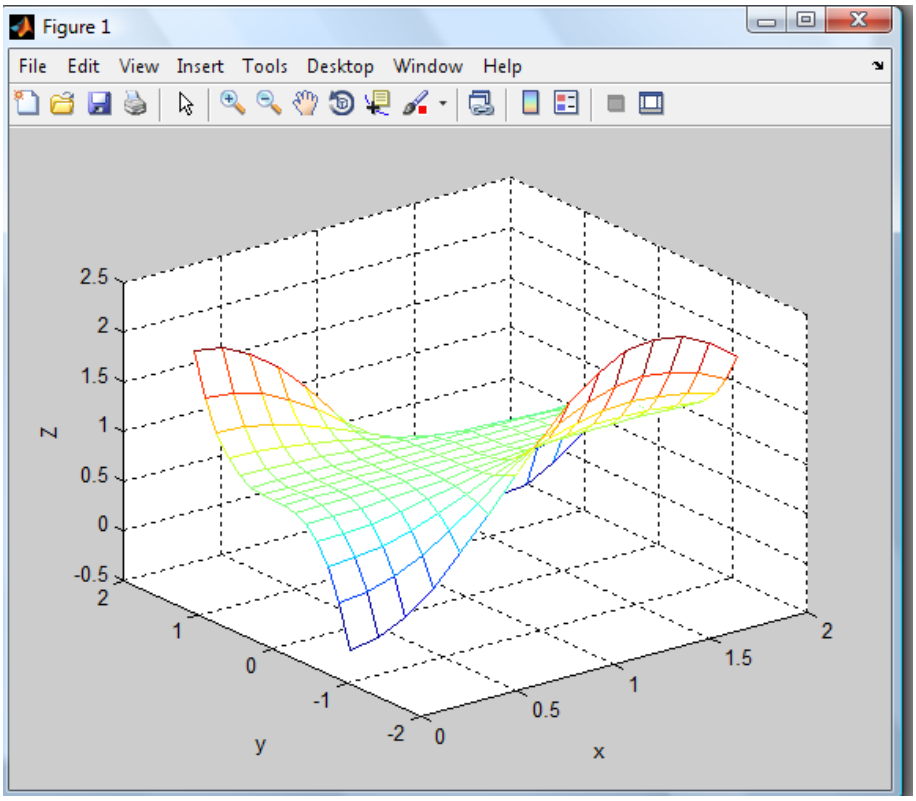
```
function k=yuzey(x,y)            % yuzey.m, iki değişkenli bir fonksiyonu hesaplar
k= cos(2*x).*y.^3+1;
```

yuzey.m adlı altprogram ile verilen z=f(x,f) fonksiyonunun $0\leq x\leq 2$, $-\pi/3\leq y\leq \pi/3$ aralığındaki entegrali aşağıdaki program satırı ile hesaplanır:

```
>>dblquad (‘yuzey’, 0, 2, -pi/3, pi/3 )
ans=
        4.1818
```

Yukarıda verilen program (altprogram kullanılmadan) aşağıda gösterildiği şekilde de yazılabilirdi.

```
x= linspace (0 , 2, 15) ; y = linspace (-pi/3, pi/3, 15);
[X,Y]=meshgrid (x, y) ; z= cos (2*X).*. ^3 +1;
mesh(X, Y, Z) , xlabel (‘ x’), ylabel (‘ y ‘), zlabel(‘ z ‘)
dblquad (‘ cos (2*x). *y.^3 +1’, 0, 2, -pi/3, pi/3 ) ;
```



Şekil 13.18

13.2.4 Üç boyutlu entegrasyon

Aşağıda verilen üç boyutlu entegral;

S=$\int_{zmin}^{zmax}\int_{ymin}^{ymax}\int_{xmin}^{xmax} f(x,y)dxdy$

(13.15)

MATLAB ortamında (Simpson veya Lobatto yöntemleri ile üç boyutlu entegral alma işlemi için) *triplequad* komutu ile hesaplanır.

V=triplequad(‘ fonksiyon ’ , xmin,xmax,ymin,ymax,zmin,zmax): fonksiyon yerine gelecek eğri denklemi , üç değişkenli olan q=f(x,y,z) şeklinde olamlıdır. Dış entegralin alt sınırı zmin, üst sınırı zmax, oratadaki entegralin alt sınırı; ymin, üst sınırı ise ymax alınmalıdır. En içteki entegralin alt sınırı xmin, üst sınırı xmax olamalıdır. Triplequad komutunu kullanmak için öncelikle f(x,y,z) eğrisi tanımlanmalıdır.

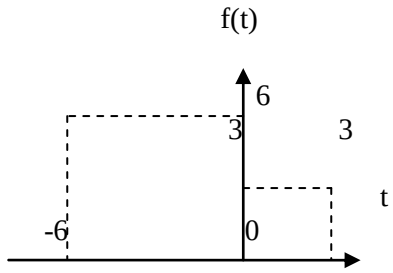
>> V= **triplequad** (‘x. ^3+60*y-3*z’ ,-1, 2, 0, 1, 2, 6);

V=

Yukarıdaki komut satırı , z =(x^3+60y)/3 denklemi ile verilen yüzeyle çevrilen hacmi,verilen x[-1;2], y[0;1],Z[2:6] aralıklarında hesaplanmaktadır.Kullanıcı isterse , yukarıdaki komut satırı içinde yer alan iki tırnak içindeki yere bir alt program ismi yazılabilir ve daha sonra bu alt program dosyası içine üç boyutlu entegre edilecek fonksiyonu yazarak da aynı işlemi yapabilir.

SORULAR

1-



Yukarıdaki şekilde verilen f(t) eğrisinin (df(t)/dt) türvini geri fark yaklaşımı ile hesaplayarak çizdiriniz.

Çözüm

```
t=-6:0.01:3; k =length(t);
for m=1:k
        if t(m) >= -6 & t(m)<=0
        y(m) = -t(m);
else
        y(m)=t(m);
        end
end
dt = diff(t) ;    dy=diff(y);   turev_y_t=diff(y)./diff(t);
plot (t (2:k) ), turev_y_t, 'k.' ) ,  grid
```

2- Bir cismin hız değişimi ,v(t) = $4t^2$ + 4t + 6 m/s olarak verilmiştir. Hareketlinin t=4. saniyedeki **a)** ivmesini dv/dt formülü ile hesaplayınız **b)** a=diff(v)./diff(t) MATLAB komutu ile hesaplayınız.

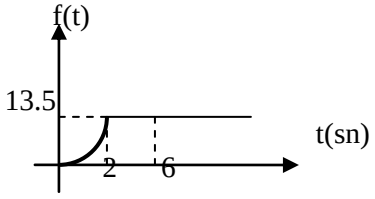
Çözüm

a) $a=\frac{dv}{dt}=\frac{d(4t^2+4t+6)}{dt}=8t+4|\sum_{t=3}=36\mathrm{m/sn}^2$ bulunur.

b)
```
>> t=0:0.1:4;
>> v=4*t.^2+4*t+6;
>> dt=diff(t); dy=diff(v);
>> a=diff(v)./diff(t);
>> ivme_4saniye=a(end)

ivme_4saniye =
```

35.6000

3) Şekil 'de f(t)=$e^{0.52*t} - 1$ eğrisi ile t ekseni arasında kalan alanı, t=[0:4] aralığında *trapz* komutunu kullanarak bulan bir m dosyası yazınız.



Çözüm

```
>> t=0:0.01:6;
>> for  k=1:length(t)
if  t(k)>=0 & t(k)<=2
f(k)=exp(0.52*t(k))-1;
else
f(k)=13.5;
end
end
>> alan=trapz(t,f)

alan =

   7.5194
```

Zaman (s)	0	1	3	5	7	9	11	13	15	17	19
Hız(m/sn)	0	26	34	69	89	145	189	190	245	390	499

4) Yukarıda zaman-hız değerleri kullanılarak hareketlinin zaman-yol-hız deerlerini bulan ve ditör ortamında yazılan MATLAB programını yazınız.

Çözüm

```
t=[0:1:19];
V=[0,26,34,69,89,145,189,190,245,390,499];
S=cumtrapz (t, V);
disp ([‘      zaman              Hiz                 Yol’ ]) ; disp( [t’ ,V’ , S] )
plot(t, V, ‘k-’, t, S, ‘k-- ’ )
```

5) f(t)=-t^3+4t^2-2t eğrisi ile t ekseni arasında t=0:4.2 saniye aralığında kalan toplam alanı hesaplayınız.

Çözüm

```
>> t=0:0.01:4.2;
>> f = -(t.^3)+ 4*(t.^2)-(2*t);
>> plot (t,f),xlabel ('t'), ylabel ('f')
>> alan=quad ('abs (-(x.^3)+4*(x.^2)-(2*x))',0,4.2)
```

alan =

113.7333

BÖLÜM 14

DİFERANSİYEL DENKLEMLERİN ÇÖZÜMÜ

MATLAB ortamında diferansiyel denklemler hem sayısal hem de sembolik (analitik) olarak çözülebilir. Sembolik ortamda diferansiyel denklem çözmek için dsolve komutu kullanılır. Bu bölümde sayısal olarak diferansiyel denklem çözümü geniş biçimde anlatılmıştır.

Diferansiyel denklemler doğrusal (lineer)ve doğrusal olmayan (nonlineer) olarak iki ayrı başlık altında incelenebilir. Diferansiyel denklemler sabit veya değişken katsayılı olabilirler. Sabit katsayılı zamana bağlı diferansiyel denklemde katsayılar zamandan bağımsızdır. Değişken katsayılı diferansiyel denklemde ise katsayılar zamanla değişir.

Doğrusal diferansiyel denklemin analitik çözümleri elde edilebilir fakat doğrusal olmayan bir diferansiyel denklemin analitik çözümüne ulaşılamayabilir. Bu durumda da bu tür diferansiyel denklemler sayısal olarak çözülmelidir.

Diferansiyel denklemlerin sayısal çözümünde kullanılan yöntemlerden ikisi Euler ve Runge-Kutta yaklaşımlarıdır. Bu yaklaşımlar,

$$g(b) = g(a) + (b-a) * g'(a) + \frac{(b-a)^2}{2!} * g''(a) + \ldots + \frac{(b-a)^n}{n!} * g^{(n)}(a) + \ldots \qquad (14.1)$$

olarak verilen Taylor seri açılımını kullanırlar. (14.1) eşitliği, x=a noktasındaki değeri bilinen bir g fonksiyonunun x=b noktasındaki değerini hesaplanır. Bu eşitlikte kullanılan g(a) ve g(b) değerleri g fonksiyonunun sırası ile a ve b noktalarındaki değerleridir. $g(a)', g''(a), \ldots, g(a)^{(n)}$ ise g fonksiyonunun sırasıile birinci, ikinci,....,n. mertebeden türevlerinin a noktasındaki değerleridir. g fonksiyonunun Taylor serisine açılabilmesi için [a b] aralığında türevinin tanımlı olması gerekir. (14.1) eşitliği ile verilen Taylor serisinin 'birinci mertebeden' açılımı;

$$g(b) = g(a) + (b-a) * g'(a) \tag{14.2}$$

olur. Taylor serisinin 'ikinci mertebeden' açılımı ise;

$$g(b) = g(a) + (b-a) * g'(a) + \frac{(b-a)^2}{2!} * g''(a) \tag{14.3}$$

ifadesi ile verilebilir. Yukarıda da görüldüğü gibi açılımlarda mertebe sayısı kadar türev bulunmaktadır. (14.1) eşitliği yerine (14.2) veya (14.3) eşitliği kullanıldığında, yüksek mertebeden türevlere ihtiyaç duyulmamaktadır. Bu yaklaşımdan dolayı ortaya çıkan hata miktarı kabul edilebilir olduğu sürece, bu yaklaşımın bilgisayarda ciddi bir zaman tasarrufu sağladığı hemen görülebilir.

Açıklama:
Birinci mertebeden diferansiyel denklemlerin entegralinde kullanılan en yaygın yöntem Runge-Kutta (R-K) metodudur. Birinci mertebeden R-K metodu birinci mertebeden Taylor açılımını, ikinci mertebeden R-K metodu ikinci mertebeden Taylor açılımını kullanır ve bu ilişki böyle devam eder. R-K metodu Euler metodunun bulunmasından sonra geliştirilmiştir. Birden çok mertebeden R-K metodu mevcuttur. Birinci mertebeden R-K metodu ile Euler metodu, aynı algoritmayı kullanır.

14.1. Runge-Kutta yaklaşımları

Yukarıda da bahsedildiği gibi kullanılan türev mertebesi arttıkça, Runge-Kutta mertebeleri de artacaktır. MATLAB'da çoğunlukla 4. mertebeye kadar Runge-Kutta yaklaşımları kullanıldığı için, 4. mertebeden yüksek Runge-Kutta yaklaşımları ele alınmamakla birlikte. daha genel bilgiler içerdiği için öncelikle 4. mertebeden Runge-Kutta yaklaşımı tanıtılmıştır.

14.1.1. Dördüncü mertebeden Runge-Kutta yaklaşımı

Kullanıcın elinde,

$$\frac{dy(x)}{dx} = f(x, y) \tag{14.4}$$

olarak verilen ve belirlenen aralıkta tanımlı bir adet (birinci mertebeden) diferansiyel denklem olsun. Bu diferansiyel denklemden yola çıkarak, y(x) eğrisini sağlayan y ve x vektörlerinin sayısal karşılığının bulunması (ve arzu edilirse y(x) eğrisinin çizdirilmesi), diferansiyel denklemin sayısal olarak çözülmesi anlamına gelir. 4. mertebeden (n=4) R-K yaklaşımında, y vektörünün (k+1). iterasyon sonundaki değeri;

$$y_{k+1} = y_k + w_1 y_1 + w_2 y_2 + w_3 y_3 + w_4 y_4 \quad (14.5)$$

Eşitliği ile hesaplanır. (14.5) eşitliğinde kullanılan k_i değerleri (h:adım aralığı olmak üzere)

$$k_1 = h * f(x_k, y_k)$$
$$k_2 = h * f(x_k + a_1 h, y_k + b_1 k_1)$$
$$k_3 = h * f(x_k + a_2 h, y_k + b_2 k_1 + b_3 k_2)$$
$$k_4 = h * f(x_k + a_3 h, y_k + b_4 k_1 + b_5 k_2 + b_6 k_3) \quad (14.6)$$

Olarak verilir. (14.4) ve (14.5) eşitliklerinin Taylor serisine açılımından;

$$b_1 = a_1 \quad (1)$$
$$b_2 + b_3 = a_2 \quad (2)$$
$$b_4 + b_5 + b_6 = a_3 \quad (3)$$
$$w_1 + w_2 + w_3 + w_4 = 1 \quad (4)$$
$$w_2 a_1 + w_3 a_2 + w_4 a_3 = 1/2 \quad (5)$$
$$w_2 a_1^2 + w_3 a_2^2 + w_4 a_3^2 = 1/3 \quad (6)$$
$$w_2 a_1^3 + w_3 a_2^3 + w_4 a_3^3 = 1/4 \quad (7)$$
$$w_3 a_1 b_3 + w_4 (a_1 b_5 + a_2 b_6) = 1/6 \quad (8)$$
$$w_3 a_1 a_2 b_3 + w_4 a_3 (a_1 b_5 + a_2 b_6) = 1/8 \quad (9)$$
$$w_3 a_1^2 b_3 + w_4 (a_1^2 b_5 + a_2^2 b_6) = 1/12 \quad (10)$$
$$w_4 a_1 b_3 b_6 = 1/24 \quad (11) \quad (14.7)$$

elde edilir. Yukarıda verilen 11 adet eşitlikte 13 adet bilinmeyen vardır. Bu eşitlikleri çözebilmek için iki adet eşitliğe daha ihtiyaç duyulur. Bu iki eşitlik için genellikle kullanılanlar;

$a_1 = 0.5;\ b_2 = 0$ (bu iki değerden farklı değer kullanan yaklaşımlar da bulunmaktadır) eşitlikleridir. Bu iki değer kullanılarak elde edilen denklem sisteminin çözümü ile;

$$a_2 = 0.5,\quad a_3 = 1,\quad b_1 = 0.5,\quad b_3 = 0.5,\quad b_4 = 0,\quad b_5 = 0,\quad b_6 = 1,$$
$$w_1 = 1/6,\quad w_2 = 1/3\quad w_3 = 1/3\quad w_4 = 1/6$$

değerleri bulunur. Elde edilen katsayılar (14.5) ve (14.6)'da yerlerine konulur ve düzenlenirse;

$$y_{k+1} = y_k + \frac{h*(f_1 + 2f_2 + 2f_3 + f_4)}{6} \quad (14.8)$$

bulunur. (14.8) eşitliğinde verilen f_1 ,.. f_4 değerleri açık olarak aşağıda gösterilmiştir;

$$f_1 = f(x_k, y_k)$$
$$f_2 = h * f(x_k + 0.5h, y_k + 0.5 * f_1)$$
$$f_3 = h * f(x_k + 0.5h, y_k + 0.5h * f_2)$$
$$f_4 = h * f(x_k + 0.5h, y_k + h * f_3)$$

(14.9)

Problem 14.1

$$\frac{dy}{dx} = f(x, y) = -0.4y - 2xy - 5x^2 - 5x + 8$$

diferansiyel denklemini ($x_{ilk} = 0$ ile $x_{son} = 3.5$ aralığında h=0.5 artışla) 4.mertebeden Runge-Kutta yöntemini kullanarak çözen MATLAB programını yazınız.

$x_{ilk} = 0$ için $y_{ilk}(0) = 1$ alınacaktır.

Çözüm

Aşağıda ana program satırları gösterilmiştir.

```
x(1)=0; xson=3.5; h=0.5;
adimsay=(xson-x(1))/h+1;
x=zeros(1,adimsay+1); y=zeros(1,adimsay+1);
y(1)=1;
M=fonkalt1('fonkalt2',x,y,h,adimsay);
x=M(:,1);
y=M(:,2); plot(x,y),xlabel('x'),ylabel('y')
```

Ana program içinde kullanılan function dosyaları:

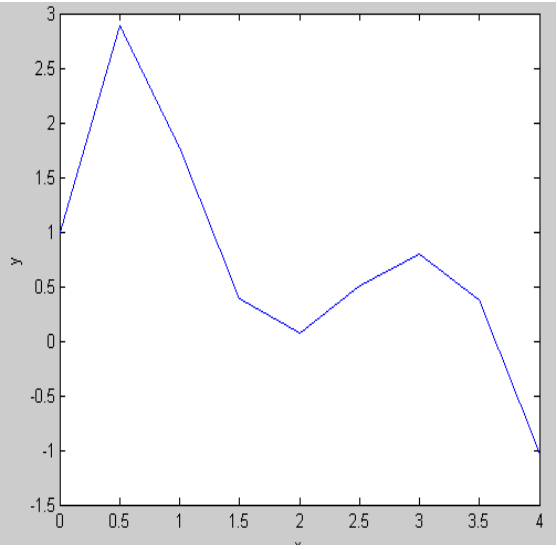
1) fonkalt1.m adlı Matlab dosyası:

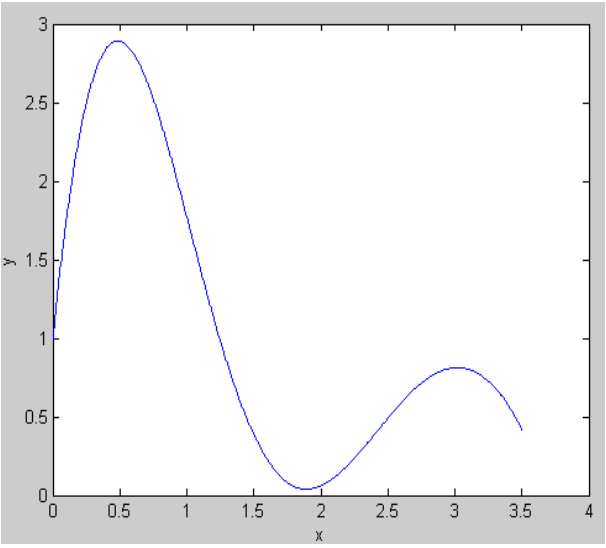
```
function M=fonkalt1(f, x, y, h, adimsay)
     for k=1:adimsay
       k1=h*feval(f,x(k),y(k));
       k2=h*feval(f,x(k)+0.5*h,y(k)+0.5*k1);
       k3=h*feval(f,x(k)+0.5*h,y(k)+0.5*k2);
       k4=h*feval(f,x(k)+h,y(k)+k3);
       y(k+1)=y(k)+(k1+2*k2+2*k3+k4)/6;
       x(k+1)=x(k)+h;
     end
     M=[x' y']; % dif.denklemin çözümü olan x,y vektörleri
```

2) fonkalt2.m adlı Matlab dosyası:

```
function f=fonkalt2(x,y)
     f= -y.*x-2*x.^3+12*x.^2-20*x+8.5;  % çözülmesi istenen
diferansiyel denklem
```

MATLAB programında h=0.5(solda) ve h=0.005(sağda) için aşağıdaki grafik elde edilir. Böylece h adım değerinin çözüm üzerindeki etkisi gösterilmiştir.





14.1.2. Euler yöntemi

f; gerçek çözüm (aranan) eğrisi olan y(x)'i temsil etmek üzere, y_a a ve h değerleri bilindiği kabulü ile, y_b (bir sonraki iterasyonda y(x) eğrisinin alacağı) değeri şekil 14.3'den;

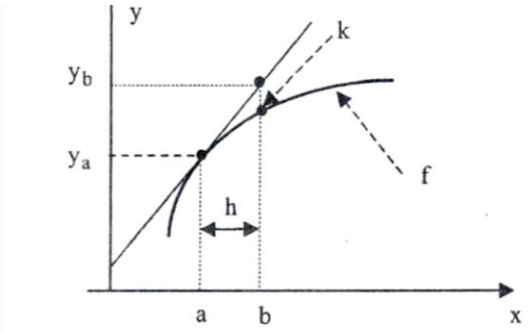
$$y_b = y_a + h * y_a' \qquad (14.10)$$

olacaktır. Şekil 14.3'de, y_b değerini hesaplamak için, x=a' da çizilen teğet çizgiden(y_a') faydalanılmıştır. Şekil 14.3'de gösterildiği gibi elde edilmesi gereken değer k olması gerekirken y_b elde edilmektedir. Böylece daha iterasyonun ilk adımında, y_b-k değerinde bir hata ile karşılaşılmaktadır. Bu hatanın küçülmesi h (adım) değerinin küçülmesine bağlı olduğu, şekil 14.3'den anlaşılmaktadır. Bu hata değeri ile bir sonraki adıma gidilecek olursa (diğer bir ifade ile y_c değeri aranırsa), şekil 14.4'den faydalanılarak;

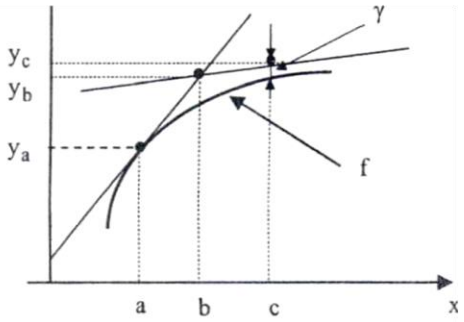
$$y_c = y_b + h * y_b' \qquad (14.11)$$

elde edilebilir. y_c değerini hesaplamak için x=b'de çizilen teğet çizgiden faydalanılmıştır. (14.10) ve (14.11) eşitlikleri diğer x noktaları için adım adım yapılarak y=f(x) sürekli fonksiyonunun tanımlanan ayrık noktalarına karşı gelen yaklaşık değerleri hesaplanır. y=f(x) fonksiyonuna ilişkin değerlerin hesabına başlayabilmek için bir ilk koşula (y_a) ihtiyaç duyulur.

Euler yöntemi çok basit bir yöntem olmakla yukarıda bahsedilen nedenlerden dolayı hassasiyeti düşüktür. Hesaplama adımı olarak adlandırılan h değeri büyük alınırsa, eğrinin ani değişim gösterdiği yerlerde önemli hatalar ortaya çıkar (bak. şekil 14.2). Şekil 14.4'de gerçek değer ile hesaplanan değer arasındaki fark γ ile gösterilmiştir. h değeri büyüdükçe y değeri artacak, γ sapması ile hesaplanan bir sonraki değer daha da hatalı bir sonuç ortaya çıkaracaktır. Yukarıda açıklanan nedenlerden dolayı Euler yönteminin yetersiz kaldığı durumlarda daha yüksek mertebeden R-K metodu kullanılır. Örneğin, dördüncü mertebeden R-K metodunda, l.,2.,3., ve 4. mertebeden türev fonksiyonları kullanılır. Şekil 14.5'de Euler yönteminin işaret akış şeması verilmiştir.



Şekil 13.3



Şekil 13.4

Problem 14.2

$$\frac{dy}{dx} = f(x,y) = -2x^3 + 10x^2 - 8x + 3$$

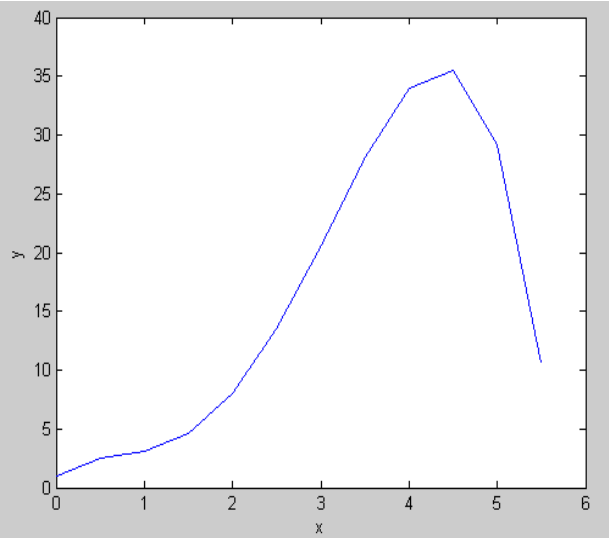
Diferansiyel denkleminin ($x_{ilk} = 0$ ile $x_{son} = 5$ aralığında h=0.5 artışla) Euler yöntemini kullanarak çözümünü bulan MATLAB programını yazınız. Burada $x_{ilk} = 0$ için $y_{ilk}(0) = 1$ alınacaktır.

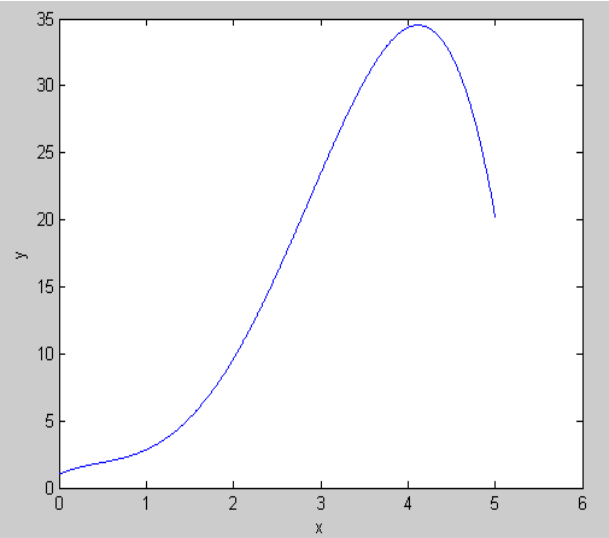
Çözüm

$$y' = f(x,y) = -2x^3 + 10x^2 - 8x + 3$$

```
x(1) = 0,  xson = 5,  y(1) = 1, h = 0.5;
hn=(xson-x(1))/h+1;                      % adım sayısı  hesaplanıyor
for k=1:hn
  f(k)= -2*x(k)^3+10*x(k)^2-8*x(k)+3;  %  çözülmesi istenen dif. denklem
  y(k+1)=y(k)+f(k)*h;                  %  euler denklemi uygulanıyor
  x(k+1)=x(k)+h;                  % bir sonraki adımdaki x değeri hesaplanıyor
end
plot(x,y), xlabel('x'), ylabel('y')
```

Aşağıda h=0.5(solda) ve h=0.05(sağda) adım değerleri için elde edilen iki ayrı çözüm çizdirilmiştir. h=0.05 için elde edilen çözüm gerçeği yansıtırken, h=0.5 için elde edilen çözüm ise (iterasyon adımı arttıkça) gerçek eğriden uzaklaşmaktadır. Birinci mertebeden R-K yöntemi ile Euler yöntemi aynı olduğundan, aşağıda ikinci mertebeden R-K yöntemi tanıtılmıştır.





14.1.3. İkinci mertebeden Runge-Kutta yöntemi

Daha önce verilen (14.5) eşitliği ikinci mertebeden Runge-Kutta yöntemi için düzenlenirse;

$$y_{k+1} = y_k + w_1 k_1 + w_2 k_2$$

(14.12)

elde edilir. 4. mertebeden Runge-Kutta yaklaşımı için yapılan işlemler burada da yapılırsa; k,

$$k_1 = h * f(x_k, y_k)$$

$$k_2 = h * f(x_k + a_1 h, y_k + b_1 k_1) \qquad (14.13)$$

elde edilir. (14.12) ve (14.3) eşitlikleri Taylor serisine açılırsa bilinmeyen 4 adet sabit için 3 adet;

$$w_1 + w_2 = 1$$

$$w_2 * a_1 = 0.5$$

$$w_2 * b_1 = 0.5$$

eşitlik elde edilir. 4 adet denklem 3 adet bilinmeyen olduğu için, bir bilinmeyenin (w_2) değeri biliniyor var sayılarak diğerleri bunun cinsinden yazılırsa;

$$w_1 = 1 - w_2$$

$$a_1 = b_1 = 0.5 / w_2$$

bulunur. Eğer w_2 = 0.5 alınırsa w_1 = 0.5, $a_1 = b_1$ =1 elde edilir. Bu sabitler için elde edilen 2. Mertebeden Runge-Kutta yaklaşımı Heun yöntemi olarak adlandırılır. Şekil 14.7'de Heun yöntemine ilişkin işaret akış şeması görülmektedir.

Eğer w_2 =1 seçilirse bu durumda w_1 = 0, $a_1 = b_1$ =0.5 elde edilir. Bu sabitler için elde edilen 2. mertebeden Runga-Kutta yaklaşımı orta nokta yöntemi olarak adlandırılır.

Problem 14.3

$$\frac{dy}{dx} = f(x, y) = -0.4y - 2xy - 5x^2 - 5x + 8$$

Diferansiyel denkleminin $x_{ilk} = 0$, $x_{son} = 4.5$ ve h=0.5 için ikinci mertebeden Runge-Kutte yöntemini kullanarak(Heun katsayıları yardımı ile) çözen MATLAB programını yazınız. $x_{ilk} = 0$ için $y_{ilk}(0) = 1$ alınacaktır.

Çözüm

Aşağıda ana program satırları gösterilmiştir.

```
x(1) = 0; xson = 4.5; h = 0.5;
adim=(xson-x(1))/h+1;     % adım sayısı hesaplanıyor
x=zeros(1,adim+1);  y=zeros(1,adim+1);  x(1)=0;     y(1)=1;
M=fonkana('fonkalt',x,y,h,adim);  x=M(:,1);  y=M(:,2);
plot(x,y,'k:'), xlabel('x'), ylabel('y');
```

Verilen ana program içerisinde kullanılan function altprogram dosyaları aşağıda gösterilmiştir.

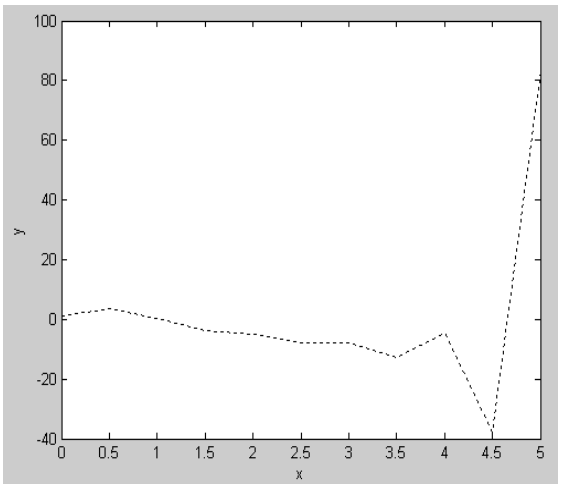
1) Fonkana.m adlı Matlab dosyası;

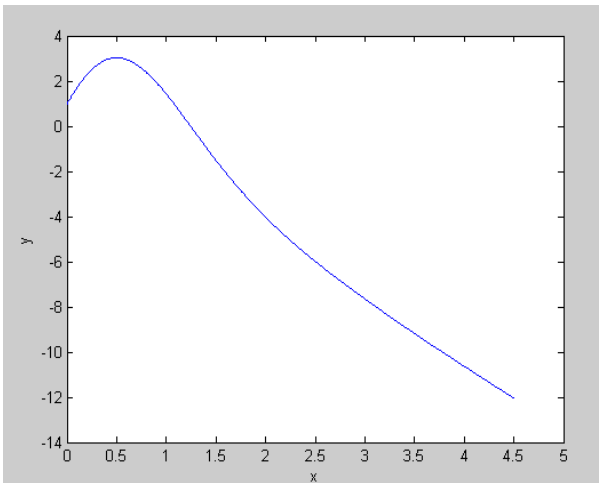
```
function M=fonkana(f, x, y, h, adim)
    w1=0.5;        % heun katsayıları giriliyor
    w2=0.5;
    a1=1;  b1=1;
    for k=1:adim
      m=x(k)+h;
      n=y(k)+h;
      k1=h*feval(f,x(k),y(k));
      k2=h*feval(f,m,n);
      y(k+1)=y(k)+w1*k1+w2*k2;
      x(k+1)=x(k)+h;
    end
    M=[x' y'];
```

2) Fonkalt.m adlı Matlab dosyası;

```
function f=fonkalt(x,y)
  f= -0.4*y-2*y*x-5*x.^2-5*x+8;  % çözülmesi istenen diferansiyel  denklem
```

Yukanda verilen MATLAB programlarının uygulanması sonunda elde edilen y=f(x) değişimi, h=0.5(solda) ve h=0.005(sağda) için çizdirilmiştir. Böylece h adım değerinin çözüm üzerindeki etkisi gösterilmiştir.





14.2. Birinci mertebeden lineer diferansiyel denklem sistemlerinin çözümü

Çözülmesi istenen diferansiyel denklem sayısı birden fazla (örneğin '2') ise;

$$\frac{dx}{dt} = f(t,x,y) \qquad \Leftarrow 1.\text{ dif denklem}$$

$$\Rightarrow \begin{cases} x(t_0) \\ y(t_0) \end{cases}; a \le t \le b$$

$$\frac{dy}{dt} = g(t,x,y) \qquad \Leftarrow 2.\text{ dif denklem}$$

eşitlikleri daha önce verilen eşitlikler kullanılarak çözülebilir. Burada, önce birinci mertebeden lineer diferansiyel denklem sistemlerinin çözümü Euler ile daha sonra R-K yaklaşımı kullanılarak çözülecektir

14.2.1. Birinci mertebeden lineer diferansiyel denklem sistemlerinin Euler yaklaşımı ile çözümü

Yukarıda verilen iki adet (f(t,x,y) ve g(t,x,y)) lineer birinci mertebeden diferansiyel denklemi $x(t_0)$ ve $y(t_0)$ ilk koşullan altında, t ekseni üzerindeki [a b] aralığı M adet eşit parçaya bölünerek;

h=(b-a)/M

ilk iterasyon adımında;

$$x_1 = x_0 + f(t_0, x_0, y_0) * h$$

$$y_1 = y_0 + g(t_0, x_0, y_0) * h$$

Olarak hesaplanır. İkinci adımda ise;

$$x_2 = x_1 + f(t_0 + h, x_0 + h, y_0 + h) * h$$

$$y_2 = y_1 + g(t_0 + h, x_0 + h, y_0 + h) * h$$

Olarak hesaplanır. Bu işlemler benze şekilde n. Adıma kadar devam eder:

$$t_{n+1} = t_n + h$$

$$x_{n+1} = x_n + h * f(t_n, x_n, y_n)$$

$$y_{n+1} = y_n + h * g(t_n, x_n, y_n); \quad \text{n=1,2,....,M+1}$$

Böylece $[t_{ilk}; t_{son}]$ aralığına karşı gelen $[x_{ilk}; x_{son}]$ve $[y_{ilk}; y_{son}]$ değerleri elde edilir.

Problem 14.4

$$\frac{dx}{dt} = f = x^2 - y; \frac{dy}{dt} = g = x - 4y$$

diferansiyel denklem sistemini t= [0:0.2] aralığında, x(0)=4 ve y(0)=2 ilk koşulları altında Euler yöntemine göre çözen MATLAB programını yazınız. Çözüm elemanlarını veriniz ve değişimi çizdiriniz (h=0.02 alınız).

Çözüm

```
t(1) = 0; tson = 0.2; h = 0.02;
adimsay=(tson-t(1))/h+1;
t=zeros(1,adimsay+1);
t(1)=0 ;
 for k=1:adimsay
    t(k+1)=t(k)+h;
 end
x=zeros(1,adimsay+1);
y=zeros(1,adimsay+1);
x(1)=4;
```

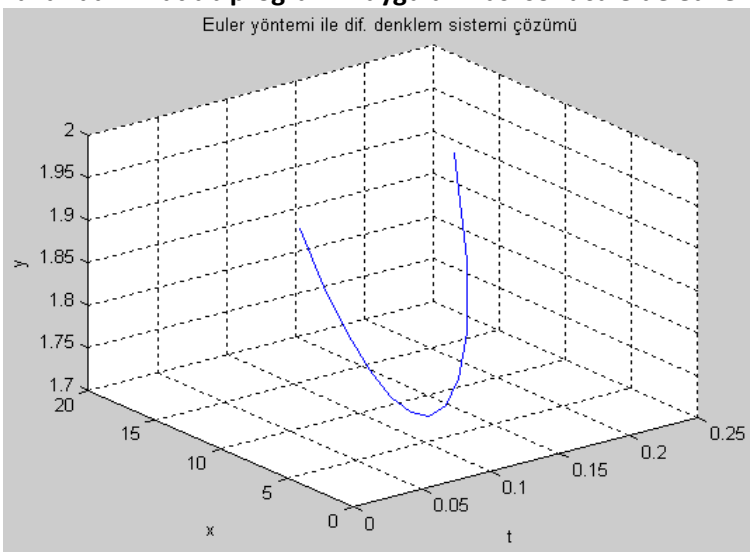
```
y(1)=2;
M=fonkcoz1('fonkcoz2','fonkcoz3',t,x,y,h,adimsay);
t=M(:,1);
x=M(:,2);
y=M(:,3);
plot3(t,x,y);xlabel('t');ylabel('x');zlabel('y');grid;
title('Euler yöntemi ile dif. denklem sistemi çözümü');
```

Yukarıdaki programın uygulanması sonucu elde edilen sonuçlar aşağıda verilmiştir.

M =

0	4.0000	2.0000
0.0200	4.2810	1.9222
0.0400	4.6101	1.8559
0.0600	4.9991	1.8012
0.0800	5.4641	1.7583
0.1000	6.0275	1.7279
0.1200	6.7214	1.7108
0.1400	7.5931	1.7086
0.1600	8.7153	1.7236
0.1800	10.2049	1.7593
0.2000	12.2605	1.8214
0.2200	15.2445	1.9189

Yukarıdaki Matlab programın uygulanması sonucu elde edilen grafik aşağıda verilmiştir.



14.2.2. Birinci mertebeden lineer diferansiyel denklem sistemlerinin 4.mertebeden R-K ile çözümü

$\frac{dx}{dt} = f(t,x,y)$ ve $\frac{dy}{dt} = g(t,x,y)$ olarak verilen diferansiyel denklem sistemi, $x_0(t) = x_0, \quad y_0(t) = y_0$ koşullan altında $a \le t \le b$ aralığında daha önce verilen (4. mertebeden) Runge-Kutta yaklaşımına benzer biçimde, (k+1). iterasyon adımında x_{k+1} ve y_{k+1} değerleri için;

$$x_{k+1} = x_k + \frac{h*(f_1 + 2f_2 + 2f_3 + f_4)}{6}$$

$$y_{k+1} = y_k + \frac{h*(g_1 + 2g_2 + 2g_3 + g_4)}{6}$$

Yazılabilir. Yakarıda verilen eşitliklerdeki f_i ve g_i değerleri;

$f_1 = f(t_k, x_k, y_k)$

$g_1 = f(t_k, x_k, y_k)$

$f_2 = f(t_k + 0.5h, x_k + 0.5h * f_1, y_k + 0.5h * g_1)$

$g_2 = g(t_k + 0.5h, x_k + 0.5h * f_1, y_k + 0.5h * g_1)$

$f_3 = f(t_k + 0.5h, x_k + 0.5h * f_2, y_k + 0.5h * g_2)$

$g_3 = g(t_k + 0.5h, x_k + 0.5h * f_2, y_k + 0.5h * g_2)$

$f_4 = f(t_k + h, x_k + h * f_3, y_k + h * g_3)$

$g_4 = g(t_k + h, x_k + h * f_3, y_k + h * g_3)$

Eşitlikleri yardımıyla bulunur.

14.3. Diferansiyel denklemlerin çözümünde kullanılan entegrasyon yöntemleri

Diferansiyel denklem çözümleri entegral alma işlemi ile bağlantılıdır. Bir fonksiyonun iki değer arasındaki entegralinin hesaplanmasında çeşitli yaklaşımlar söz konusudur. Bunlar; tek basamaklı yöntemler, çift basamaklı yöntemler, dıştan ara değer bulma yöntemleri ve sıkı (stiff) sistem yöntemleridir.

Tek basamaklı yönteme örnek olarak Euler ve Runge-Kutta verilebilir. Euler yönteminde bir hesaplama aralığında yalnızca bir adet türev hesabı gerekir. Bu yaklaşımla h değerinin küçük olması, sonucun doğruluğu açısından çok önemlidir. Daha hassas ve doğru çözüme ulaşmak için Taylor açılımında daha fazla terimin hesaba katılması gerekir. Bunun anlamı ise, daha yüksek mertebeden türevlerin hesaplamalara dahil edilerek sonuçta işlemlerin zorlaşmasıdır. Runge-Kutta yöntemi bir taraftan Taylor dizisindeki ardışık diferansiyel hesaplamalardan kaçınan fakat aynı zamanda yapılan hataları oldukça azaltan bir yöntemdir. R-K yöntemi yüksek mertebeden türevlerin hesaplanması yerine bir takım ağırlık katsayıları kullanmaktadır. R-K yaklaşımı tek basamaklı yöntemler içinde en çok kullanılan yöntemdir. Tek basamaklı yaklaşımlarda (özellikle 4.mertebeden R-K yönteminde) tek bir hesaplama aralığında birden fazla türev değerlendirmesi gerektiğinden hesaplama ekonomik olmamaktadır.

Çok basamaklı entegrasyon yöntemlerinde her bir hesaplama aralığında daha önceki hesaplama adımından elde edilen sonuçlar kullanılır. Böylece daha hızlı ve verimli bir hesaplama yapılabilir. Çok basamaklı yaklaşımlar birden fazla noktadaki çözümlerin bilinmesini zorunlu kılar. Çok basamaklı yaklaşımlar kendi içlerinde açık tip ve kapalı tip olmak üzere ikiye ayrılırlar. Açık tip çok basamaklı entegrasyon yöntemine örnek olarak Adams-Bashfort yöntemi, kapalı'tipe örnek olarak ise Adams-Moulton yöntemi verilebilir. Bu yöntemler aynı zamanda kestirme-düzeltme yöntemlerine de örnek oluşturmaktadır.

Değerleri birbirlerine göre çok farklı olan özdeğer, özvektör ve zaman sabitlerine sahip olan sistemler sıkı (stiff) sistemler olarak adlandırılır. Runge-Kutta yöntemi gibi bazı yöntemlerde sayısal hesaplama kararsızlıkları ile karşılaşılır. Sıkı sistem algoritmalarına sahip yöntemlerde bu kararsızlıklar ortadan kaldırılır. Sıkı yönteme örnek olarak Gear yaklaşımı verilebilir. MATLAB içinde yer alan SIMULINK programında Gear yaklaşımı mevcuttur.

Entegrasyon yöntemlerinde ortaya çıkan hesaplama hatalarının sebebi büyük oranda seçilen hesaplama aralığından kaynaklanmaktadır. Türevlerin özellikle yavaş değişim gösterdiği diferansiyel denklem çözümlerinde h hesaplama adımı büyük seçilmelidir. Türevlerin hızlı değişim gösterdiği durumlarda ise hesaplama adımı küçük seçilmelidir. Böyle bir yaklaşım, iterasyonun daha hızlı bitmesini temin eder.
Diferansiyel denklem çözüm yöntemleri yukarıda belirtilen nedenlerden dolayı 'uygun hata değeri seçimi' problemine dönüşmektedir. Hata değerinin uygun seçilmesi, kullanıcının kullanılan model hakkında bazı temel bilgilere sahip olmasını gerekli kılmaktadır.
Kullanıcı açısından diğer önemli bir mesele de entegrasyonun başlangıç aralığının iyi tanımlanabilmesidir. Bazen daha uygun bir başlangıç hesap aralığı seçimi ile bilgisayar bellek yükünü azaltmak mümkün olabilir.

Diferansiyel denklemin sayısal çözümünde hata değeri her iterasyon adımda büyüyor ise kullanılan yöntem 'kararsızdır' denir. Hata başlangıçtan itibaren sönümlü dalga biçiminde salınıyor ise yöntem 'kararlıdır' denir. Bazen bir çözüm h değeri küçültülerek de kararlı yapılabilir ise de bu yaklaşım genel bir uygulama değildir. Bazen h'nin çeşitli değerleri aynı problem için çözülerek uygun h değeri aranır. Ardışıl entegrasyon ile Picard yaklaşımında seriye açılım yapılır ve hata değeri artar. Deneme-düzeltme yöntemlerinde birden fazla sayıda noktanın bilinmesi gerekir ve hesaplama kendiliğinden başlayamaz. Adım uzunluğu göz önüne alınmadığından kararsızlık ortaya çıkar fakat her adımda hata kontrolü yapmak mümkündür. R-K yöntemi genelde 'kararlı' yöntemdir. Kendi kendilerine başlar ancak hesaplama zamanı deneme-düzeltme yöntemlerinden daha uzundur. Çok basamaklı yöntemlerde kararlılık yönteminin tipine bağlı olarak değişir.

14.4. MATLAB komutları ile diferansiyel denklem çözümü

Diferansiyel denklemlerin MATLAB ortamındaki sayısal çözümlerinde ode23, ode45, ode118, ode15s, ode23s, ode23t, ode23tb, odel5i olarak adlandırılan komutlar kullanılır. ode23 ve ode45 komutları Runge-Kutta yaklaşımını kullanır. ode23, ikinci ve üçüncü mertebeden Runge-Kutta yaklaşımını kullanırken, ode45 ise dördüncü ve beşinci mertebeden R-K yaklaşımını kullanır, ode113,1 mertebeden 14. mertebeye kadar değişen değerli açık ifadeli Adams kestirimci-düzeltici yöntemlerin yeni yorumu olan bir algoritmayı koşturur. ode23s, 2. ve 3. mertebeden doğrusal olarak kapalı ifadeli R-K yaklaŞımının stlff halidir, ode15 s ise 1.mertebeden 5.mertebeye kadar değişen değerli kapalı ifadeli çok basamaklı yöntemlerin yeni bir türüdür, ode15i ise implisit türden diferansiyel denklemleri çözer. ode23 komutu gecikmeli diferansiyel denklemleri sabit gecikmelerle çözmek için kullanılır. bvp4c komutu ise sınır değer problemlerini Collocation metodu kullanarak çözer. pdepe komutu ile 1 boyutlu sınır ilk değer problemlerini temsil eden parabolik ve eliptik kısmi diferansiyel denklemler çözülür.
Runge-Kutta gibi yöntemler sabit zaman artımları ile sonuca giderken, Adams, Gear gibi yöntemler her adımda uygun zaman artımını sistemin davranışını gözeterek ayarlarlar. Bu özellikler MATLAB-toolbox kullanıldığında yöntem seçerken çok önem kazanırlar.
ode komutuna eklenen s harfi, sıkı (stiff-ani değişimli) anlamına gelmektedir, s uzantılı ode komutları, sıkı olmayan yöntemlere de uygulanabilir olmasına rağmen verimli olmazlar, s içermeyen ode komutları sıkı problemlere uygulanabilirlerse de hesaplama adımı çok küçük olduğundan burada da verim elde edilemez. ode23tb komutu 2. ve 3. mertebeden R-K yaklaşımı ile sıkı denklemleri çözmek için kullanılır. ode23t komutu ılımlı (orta düzey) sıkı diferansiyel denklemleri trapez kuralına göre çözer. Yukarıda bahsedilen komutlar hem doğrusal hem de doğrusal olmayan diferansiyel denklem çözümü için kullanılabilirse de doğrusal diferansiyel

denklem çözümünde MATLAB-The control system toolbox- program paketi içinde yer alan 1sim, initial, step gibi komutlan kullanmak daha hızlı çözüm sağlamaktadır.

14.5. Diferansiyel denklemlerin çözümünde kullanılan MATLAB komutlarının tanıtımı

$$\frac{dy}{dt} = f(t, y)$$

ifadesi birinci mertebeden lineer olmayan bir diferansiyel denklem olsun. (14.14) ifadesinin sayısal çözümünü sağlayan ode komutunun matlab komut yapısı aşağıda verilmiştir:

```
[t,y] =ode23 ('fonksiyon adı',[t0  tson] ,y0, secenekler, pl,p2)
```

ode komutunda kullanılan ifadelerin tanımları aşağıda gösterilmiştir:

fonksiyon adı: Entegre edilmesi gereken diferansiyel denklem MATLAB editör ortamında .m uzantılı bir fonksiyon dosyası adı verilerek kaydedilir. Bu dosyanın adı yukarıda fonksiyon adı olarak gösterilen ve tırnak içinde belirtilen yere yazılır (m uzantısı yazılmaz). Dosyanın adı mutlaka iki tırnak işaretinin arasına yazılmalıdır. [t,y] ile gösterilen çıkış değerlerinde y' nin satır sayısı t'nin boyutu ile aynıdır. Diferansiyel denklem birinci mertebeden ise y (:, 1) vektörü y(t) değişimini, y (:, 2) vektörü ise dy/dt değişimini verir. Diferansiyel denklem daha yüksek mertebeden ise y (:, 3); d^2y/dt^2, y (:, 4); d^3y/dt^3,.... şeklinde diğer yüksek mertebeden değişimleri içerir.

ode komutu ile birlikte kullanılacak fonksiyon dosyası;

```
function dy_dt = 'dosya adı'(t,y)
```

satırı ile başlar ve çözülmesi istenen diferansiyel denklemin MATLAB formatında yazımı ile devam eder.

t0 : diferansiyel denklemdeki bağımsız değişkenin alt sınır değeridir. Bu değişken 'zaman' olmak zorunda değildir, başka değişken de olabilir.

t son : diferansiyel denklemdeki bağımsız değişkenin üst sınır değeridir. Bu değişken 't=zaman olmak zorunda değildir.

y0 : diferansiyel denklemin başlangıç koşullarını temsil eden vektördür. Eğer problem sınır değer problemi ise y0 ile sınır değerler belirtilir, n. dereceden bir diferansiyel denklem için y0'ın boyutu n olmalıdır.

seçenekler: diferansiyel denklem entegre edilirken ekrana yazdırılacak mesajlar, maksimum adım sayısı, entegrasyon işleminin hata sınırlan (odeset komutu içinde belirtilen) gibi özellikler yazılır.

```
[t y] = ode23 (.....)
```

Yukarıda görülen t ve y ifadeleri diferansiyel denklemin matlab ortamında çözümünden elde edilirler. Diğer bir ifade ile diferansiyel denklemi sağlayan ikililerin kümesidir, t bir vektör, y bir matristir. Yukarıdaki ode komutunun sol tarafında yer alan t argümanı bir sütun vektörü şeklinde zaman değerlerini temsil eder. y ise durum değişkenleri matrisini temsil eder. y matrisi, durum değişkenlerinin sayısına eşit değerde sütuna sahiptir. `y(:,1)` vektörü y(t) değişimini, `y(:,2)` vektörü ise dy/dt değişimini verir. Diferansiyel denklem daha yüksek mertebeden ise `y(:, 3);` d^2y/dt^2, `y(:, 4);` d^3y/dt^3,.... şeklinde diğer yüksek mertebeden değişimleri içerir.

Problem 14.6

Aşağıda verilen birinci mertebeden diferansiyel denklemin y(0)=l başlangıç koşulu altında 0 ve 3 sınır değerleri arasındaki sayısal çözümünü ode komutu yardımı ile çizdiriniz.

$$y' = \frac{dy}{dt} = g(t,y) = t^2 - y$$

Çözüm

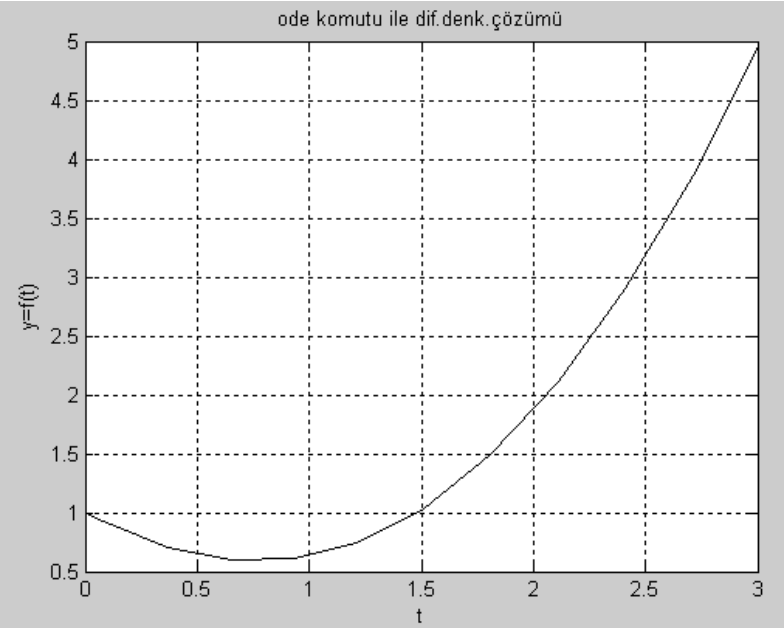
Ode23 komutu kullanılarak g(t,y)'nin sayısal çözümü bulunmak istenir ise önce g(t,y) diferansiyel denklemi function komutu kullanılarak bir alt program dosyasına yazılmalıdır(alt1.m):

```
function dy_dt=alt1(t,y)
dy_dt=t.^2-y;
```

Aşağıda ise cozede1.m adlı altprogramı çağıran ana program verilmiştir.

```
%aşağıdaki satır g(x,y) diferansiyel denkleminin sayısal çözüm noktalarını verir.
[t, y]=ode23('alt1', [0 3],1);
plot(t,y,'k-'); title('ode komutu ile dif.denk.çözümü');
xlabel('t'),ylabel('y=f(t)'),grid;
```

Aşağıdaki şekilde yukarıda verilen matlab programından elde edilen çıkış değerleri gösterilmiştir.



Problem 14.7

$$\frac{di(t)}{dt} = -15*i(t) + 2*\cos(2*t) + t$$

Yukarıda verilen sabit katsayılı lineer diferansiyel denklemin çözümünü, t = [0:3] saniye zaman aralığında ve i(0)=0 ilk koşulu altında, ode45 komutu yardımı ile (i(t)) çizdiriniz.

Çözüm

Yukarıda elde edilen diferansiyel denklem cozede2.m adlı program içine yazılırsa,

```
function di_dt=cozede2(t,a)
di_dt=-15*a+2*cos(2*t)+t;
```

elde edilir. Bu altprogramı çağıran MATLAB programı aşağıda verilmiş ve bu programda ode45 komutu kullanılmıştır. Programın çalıştırılması i(t) değişimi şekilde verilmiştir.

```
[t, a]=ode45('cozode2',[0 3],0);
plot(t,a);
title('dif.denklem çözümü');
xlabel('t'),ylabel('i=i(t)')
grid
```

14.6. Yüksek mertebeden sabit katsayılı bir diferansiyel denklemin ode komutu ile çözümü

n. mertebeden sabit katsayılı bir diferansiyel denklemin, yukarıda anlatılan yaklaşımlardan faydalanılarak çözülebilmesi için, değişken dönüşümü yapılarak n adet birinci mertebeden diferansiyel denklem sistemine dönüştürülmesi gerekir, n. mertebeden bir diferansiyel denklem;

$$c_n \frac{d^n y(t)}{dt^n} + c_{n-1} \frac{d^{n-1} y(t)}{dt^{n-1}} + c_{n-2} \frac{d^{n-2} y(t)}{dt^{n-2}} + \ldots\ldots\ldots + c_0 y(t) = f(t)$$

(14.15)

Veya

$$y^{(n)}(t) = (\frac{c_{n-1}}{c_n} y^{(n-1)}(t) + \frac{c_{n-2}}{c_n} y^{(n-2)}(t) + \ldots + \frac{c_0}{c_n} y(t)) + \frac{1}{c_n} f(t)$$

$$x_n'(t) = y^{(n)}(t) = a_{n-1} y^{(n-1)}(t) + a_{n-2} y^{(n-2)}(t) + \ldots + a_0 y(t) + k * f(t)$$

$$x_{n-1}'(t) = y^{(n-1)}(t)$$

$$\vdots$$

$$x_1'(t) = y'(t)$$

$$y(t) = x_1(t), \quad y'(t) = x_1(t), ve \quad y''(t) = x_1(t)$$

$$y = y(0) = x_1(0) = -1, \quad y'(0) = x_2(0) = 1, \quad y''(0) = x_3(0) = 2$$

$$x - 4x' - 5 = t$$

$$x''' + x = 2t$$

$$x'(0) = 1 \quad \text{ve } x''(0) = 3$$

$$c_n y^{(n)}(t) + c_{n-1} y^{(n-1)}(t) + c_{n-2} y^{(n-2)}(t) + \ldots + c_0 y(t) = f(t)$$

(14.6)

Olarak verilsin. (14.16) eşitliği tekrar düzenlenirse;

$$y^{(n)}(t) = (\frac{c_{n-1}}{c_n} y^{(n-1)}(t) + \frac{c_{n-2}}{c_n} y^{(n-2)}(t) + \ldots + \frac{c_0}{c_n} y(t)) + \frac{1}{c_n} f(t)$$

$$y^{(n)}(t) = a_{n-1} y^{(n-1)}(t) + a_{n-2} y^{(n-2)}(t) + \ldots + a_0 y(t) + k * f(t)$$

(14.17)

Elde edilir. Değişken dönüşümü yapılarak;

$$x_1(t) = y(t)$$

$$x_2(t) = y^{(1)}(t)$$

$$\vdots$$

$$x_n(t) = y^{(n-1)}(t)$$

(14.18)

Elde edilebilir.(14.18)'de verilen eşitliklerin her iki tarafı t'ye göre türetilirse;

$$x_n'(t) = y^{(n)}(t) = a_{n-1} y^{(n-1)}(t) + a_{n-2} y^{(n-2)}(t) + \ldots + a_0 y(t) + k * f(t)$$

$$x_{n-1}'(t) = y^{(n-1)}(t)$$

$$\vdots$$

$$x_1'(t) = y'(t)$$

$$x_1(t) = y(t)$$

(14.19)

elde edilir. Böylece (14.15)'de verilen n. mertebeden bir adet diferansiyel denklem (14.19)'da görüldüğü gibi birinci mertebeden n adet diferansiyel denkleme dönüşmüştür. Aşağıda verilen problem çözümü dikkatlice incelenmelidir.

Problem 14.8

$y''' + 5y'' - y = 4$

a) Yukarıda verilen 3. Mertebeden diferansiyel denklemi, $y = y(t) = x_1(t), \quad y' = x_2(t), \quad y'' = x_3(t)$ ifadelerini kullanarak, birinci mertebeden üç adet diferansiyel denkleme dönüştürünüz.

b) elde edilen üç adet birinci mertebeden deiferansiyel denklemi t=[0 2] aralığında ve $y = y(0) = x_1(0) = -1, \quad y'(0) = x_2(0) = 1, \quad y''(0) = x_3(0) = 2$ ilk koşulları altında ode45 komutu ile MATLAB ortamında çözünüz, $y(t) = x_1(t), \quad y'(t) = x_1(t), ve \quad y''(t) = x_1(t)$ eğrilerini aynı çizim ekranı üzerinde çizdiriniz.

Çözüm

a)

$$x'_3 = -5x_3 + x_1 + 4;$$

$$x'_2 = x_3;$$

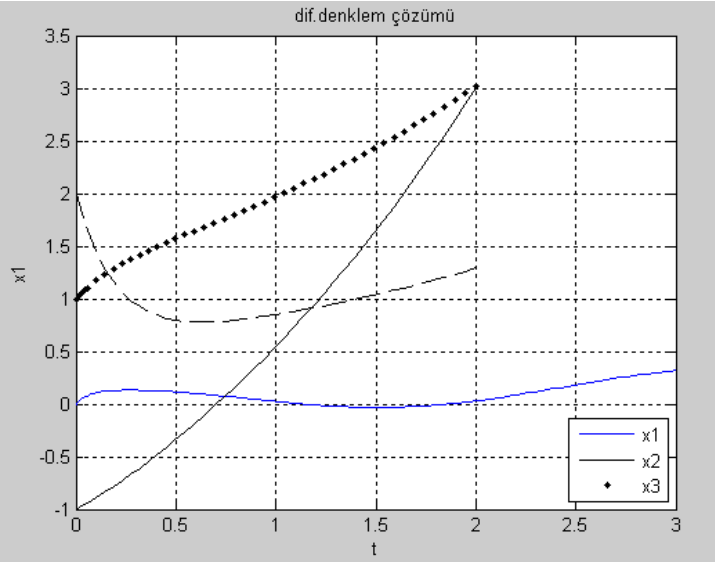
$$x'_1 = x_2$$

b)

```
x0=[-1 1  2];
[t x]=ode45('cevap2008',0,2, x0)
hold on
plot(t,x(:,1),'k-'),xlabel('t'),ylabel('x1'),grid on
plot(t,x(:,2),'k.')
plot(t,x(:,3),'k--')
legend('x1','x2','x3',4)
```

Alt program dosyası

```
Function turevler=cevap2008(t,x)
Turevler=[x(2);x(3);-5*x(3)+x(1)+4]
```



14.7. Diferansiyel denklemlerin dsolve komutu ile çözümü

dsolve komutu adi diferansiyel denklemlerin sembolik (harfleri kullanan) çözümünü veren bir MATLAB uygulamasıdır ve symbolic toolbox içinde yer alır .Bu komut içinde (d/dt) türevi D harfi ile temsil edilir. Bu komut kullanılarak yapılan diferansiyel denklem çözümlerinde eğer ilk koşullar kullanılmaz ise elde edilen çözümde sabitler de yer alır. Komut içinde ilk koşullar da kullanılırsa, elde edilen çözüm içinde sabitler yer almaz. Bu komut içinde değişkenler farklı seçilmek sureti ile birden fazla diferansiyel denklem aynı anda çözülebilir. Her diferansiyel denklem, komut satırında, yine tırnak içinde ayrı ayrı yazılmalıdır.

Problem 14.10

$x'''-4x'-5=t$ diferansiyel denklemini dsolve komutunu kullanarak çözünüz.

Çözüm

```
>> x_t=dsolve('D3x-4*Dx-5=t')
x_t=
        1/2*exp(2*t)*C2-1/2*exp(-2*t)*C1-1/8*t^2-5/4*t+C3
```

Elde edilir.

14.8. Doğrusal diferansiyel denklemlerin 1sim komutu ile çözümü

Control system toolbox' ın yapısı içinde yer alan 1sim, impulse, step, initial gibi MATLAB komutlan kullanılarak doğrusal diferansiyel denklemlerin çözümü yapılabilmektedir. Bu komutların kullanılabilmesi için verilen diferansiyel denklemin 'durum denklemi' yada 'transfer fonksiyonu' biçiminde yazılması yeterlidir. Eğer diferansiyel denklem takımı (alt indisler matris boyutlarını göstermektedir):

$$\frac{d}{dt}x(t)_{n*1}=a_{n*n}x(t)_{n*1}+b_{n*m}u(t)_{m*1}$$

(14.20)

$$y(t)_{k*1}=c_{k*n}x(t)_{n*1}+d_{k*m}u(t)_{m*1}$$

(14.21)

şekline getirilebilmiş ise 1sim komutunu kullanarak x(t)-durum değişken değerlerini- ve y(t)-çıkış değerlerini- bulmak mümkündür.

```
[y  x]=1sim (a,b,c,d,u,t,x0)
```

Yukarıda verilen MATLAB komutu içinde yer alan ve (14.20) ve (14.21) eşitliklerinde kullanılan değişkenler aşağıda tanıtılmıştır. Yukarıda verilen matris boyutlarına ilişki tanımlar aşağıda verilmiştir:

n : durum değişkenlerinin sayısı
m : giriş değişkenlerinin sayısı (kaynak sayısı)
k : çıkış değişkenlerinin sayısı

Yukanda verilen matrislerin tanımları ise aşağıda verilmiştir:

y : çıkış değerleri vektörü
x : durum değişkeni vektörü
a : durum değişkeni katsayılar matrisi
b : giriş fonksiyonu katsayılar matrisi
c : durum değişkeni katsayılar matrisi
d : giriş fonksiyonu katsayılar matrisi
u : giriş fonksiyonlan vektörü.Bu matrisin sütun sayısı giriş fonksiyonu kadar olurken, satır sayısı ise 't' zaman vektörü eleman sayısı kadar olmalıdır. Bu amaçla length komutu da kullanılabilir. Bir adetten fazla giriş işareti alan sistemlerde giriş sayısı u matrisinin sütun sayısına, length (t) ise u matrisinin satır sayısına eşit olmalıdır, t süresi 0: Δt: t_{son} formunda belirlenmelidir.

x0 : durum değişkenlerinin başlangıç değerlerini belirleyen sütün vektörüdür. Boyutu durum değişkenleri sayısına eşittir. Eğer bu vektör elemanları verilmemiş ise MATLAB, x0 vektörünü sıfır olarak alır.

```
>> 1sim (a,b,c , d , u , t,x0)
```

komutu kullanılır ise 1sim komutu y(t) değişimini çizdirir. Aynı çizim ekranı üzerinde **silik olarak** u(t) fonksiyonu da yer alır. Eğer 'x' durum değişkenlerinin değeri araştırılmıyor ise yukarıdaki komut;

```
>>y =lsim ( a , b, c, d, u, t)
```

şeklinde de kullanılabilir (bu durumda otomatik çizim yapmaz).

Problem 14.13

$y'' + 5y' - 6y = -2e^{-3t}$, y′=1; y(0)=2

a) Diferansiyel denklemini $x_1(t)$; $x_2(t) = y'$ dönüşümünü kullanarak 1sim komutu ile çözebilmek için durum uzayı formuna getiriniz. A,B,C,D matrislerini elde ediniz.

b) Yazacağınız bir program ile ve `lsim` komutunu kullanarak, y(t) eğrisini [0 6] saniye aralığında çizdiriniz.

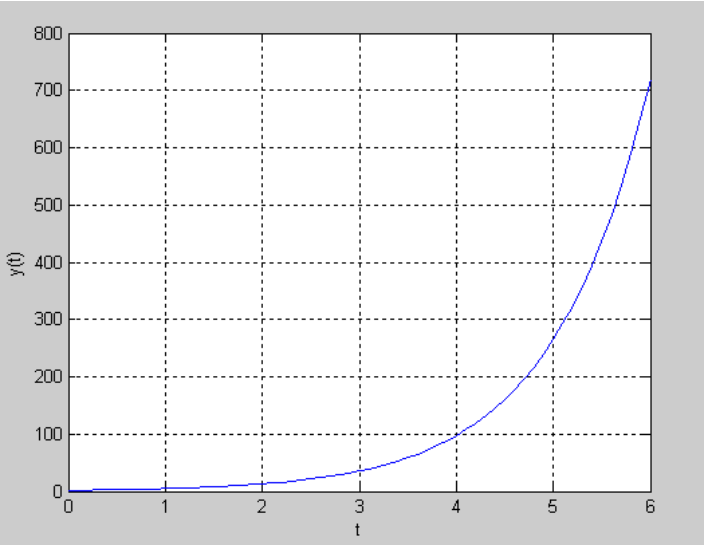
Çözüm

a)

$$x_2' + 5x_2 - 6x_1 = -2e^{-3t} \Rightarrow x_1' = x_2;\ y=x_2$$

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 6 & -5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \end{bmatrix} e^{-3t}; \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]e^{-3t}$$

b)
```
a=[0  1;6  -5]; b=[0;-2]; c=[1 0]; d = 0;
x0 = [2  1];  t = 0:0.1:6; u= exp(-3*t);% bir girişli sistem
[y x]=lsim(a,b,c,d,u,t,x0); % y çıkışı temsil eder. Bu problemde y=
olmaktadır
plot(t,x(:,1))
xlabel('t'),ylabel('y(t)'),grid
```



Problem 14.14

Seri R,L,C devresi $v(t) = \sqrt{2} * 10 * \cos(2 * \pi * f * t + \beta)$ **volt olan alternatif gerilim kaynağı tarafından beslenmektedir. Devreden akan i(t9 akımının değişimi t=[0: 2] msn aralığında çizdiriniz.** $\beta = 0.5 radyan$ i'(0)=1, i(0)=2 **alınız. (R=2 ohm, L=0.5 henry, V=4 farad, f=50 mili hz)**

Çözüm

Seri R,L,C devresinde Kirchoff gerilim yasası uygulanır ise;

$$v(t) = \sqrt{2} * 10 * \cos(2 * \pi * 50 * 10^{-3} * t + 0.5) = L\frac{di(t)}{dt} + \frac{1}{C}\int i(t)dt + Ri(t)$$

Elde edilir. Eşitliği integralden kurtarmak için her iki tarafın t'ye göre türevi alınır ise;

$$\sqrt{2}*10*2*\pi*50*10^{-3}*\sin(2*\pi*50*10^{-3}*t+0.5)=0.5*\frac{d^2i(t)}{dt^2}+0.25*\frac{di(t)}{dt}+2i(t)$$

Elde eldir. Daha basit olarak ifade edilirse;

$$-4.44*\sin\left(0.3142*t+0.5\right)=0.5*i''+0.25*i'+2*i$$

$$-8.88*\sin(\left(0.3142*t+0.5\right)=i''+0.5*i'+4*i$$

Elde edilir. İkinci mertebeden deferansiyel denklem değişken dönüşümü yardımı ile birinci mertebeden iki adet diferansiyel denkleme dönüştürülebilir:

$i'=x_{2;}$

$i=x_1$(*çıkış* fonksiyonu);alınırsa

$i''=x_{2;}'$ $i'=x_{1;}'$ $i=x_1$

Elde edilir. Yukarıda verilen diferansiyel denklem, böylece iki adet birinci mertebeden diferansiyel denkleme dönüştürülmüş olmaktadır:

$$x'=-0.5*x_2-4*x_1-8.88*\sin(2*\pi*50*10^{-3}*t+0.5)$$

$$x'=x_2$$

Bulunan denklemler durum denklemleri ve transfer fonksiyonu cinsinden yazılırsa;

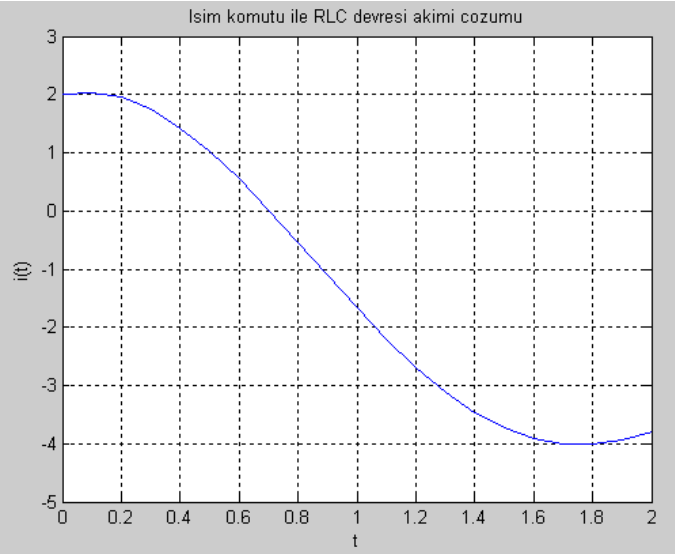
$$\frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}0 & 1\\-4 & -0.5\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+\begin{bmatrix}0\\-8.88\end{bmatrix}\sin(2*\pi*50*10^{-3}*t+0.5)$$

Not:u(t)=$\sin(2*\pi*50*10^{-3}*t+0.5)$ giriş fonksiyonudur

$$y=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+\begin{bmatrix}0\end{bmatrix}u(t);\qquad x0=[x_1(0)\quad x_2(0)]=[2\quad 1]$$

Elde edilir. Elde edilen bu matris denklemlerinin çözüldüğü MATLAB programı aşağıda verilmiştir. Program sonunda elde edilen akım-zaman değişimi ise şekilde gösterilmiştir.

```
a=[0  1;-4  -0.5]; b=[0 ; -8.88]; c=[1  0]; d=0;
x0= [2  1];
t = 0:0.1:2;   % akım değişimi 0:2 saniye arasında incelenmektedir
u = sin(2*pi*50*0.001*t+0.5); % bir girişli sistem
y =  lsim(a, b, c, d, u, t, x0);
plot(t,y);  % C vektörüne bakıldığında çıkışın (y), x1 olduğu görülmektedir
title('lsim komutu ile RLC devresi akimi cozumu');
xlabel('t');ylabel('i(t)');grid;
```



Problem 14.18

Doğru gerilim kaynağından beslenen seri R,L,C devresinde kaynak gerilimi; E=2 Volt, R=40 ohm, L=0.1 Henry, C=10^{-7} Farad olarak verilmektedir. Kapasitenin t=0 anında uçlan arasındaki gerilim değeri $v_c(0) = 0$ Volt, devre akımı i(0)=0 Amper olduğuna göre $v_c(t)$ ve i(t) değişimlerini t=[0;0.03] sn aralığında çizdiriniz.

Çözüm

Seri R,L,C devresine Kirchoff gerilim yasası uygulanırsa;

$$E = R * i(t) + \frac{L di(t)}{dt} + V_c(t)$$ Elde edilir.

Kapasitenin tanım bağıntısı;

$$i(t) = C \frac{dv_c(t)}{dt}$$

Olduğuna göre yukarıda verilen iki adet dif. Denklem aşağıdaki gibi yazılabilir.

$$\frac{d}{dt}\begin{bmatrix} v_c \\ i \end{bmatrix} = \begin{bmatrix} 0 & 1/C \\ -1/L & -R/L \end{bmatrix}\begin{bmatrix} v_c \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ E/L \end{bmatrix} u(t)$$

$$v_c = y = cikis = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_c \\ i \end{bmatrix} + 0 * u(t)$$

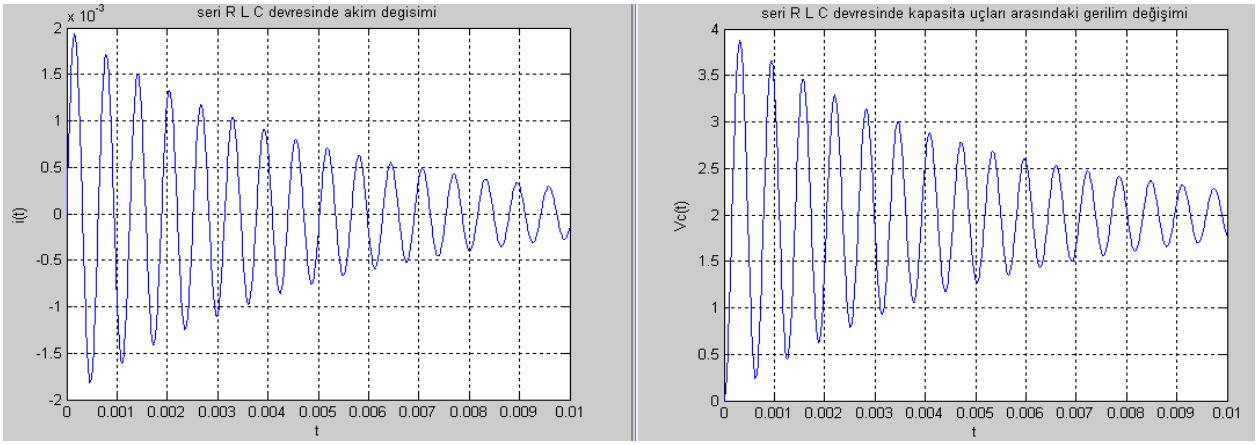
x vektörü için

$x = \begin{bmatrix} v_c \\ i \end{bmatrix}$ yazılabilir. Verilen dif. Denklem aşağıdaki program ile çözülebilir.

E=2;R=40;L=0.1;C=10^-7;

```
a = [0  1/C  ;-1/L  -R/L];b = [0; E/L], c = [1   0]
d = 0; x0 = [0  0];t = 0:0.00001:0.01;
u = ones(1,length(t)); % bir girişli sistem
[cikis x] =  lsim(a,b,c,d,u,t,x0)
figure(1)
plot(t,x(:,1)),xlabel('t'), ylabel('Vc(t)'), grid;
title(' seri R L C devresinde kapasita uçları arasındaki gerilim değişimi ');
figure(2)
plot(t,x(:,2)),xlabel('t'), ylabel('i(t)'), grid
title('seri R L C devresinde akim degisimi'),
disp(['      t        vc         i']);
disp([t',x]) ;
```

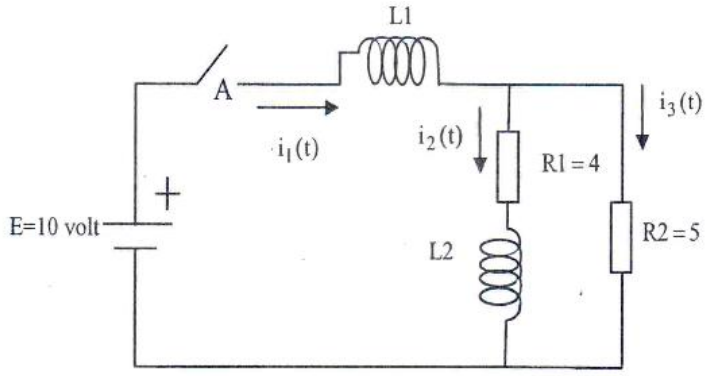
Programın çalıştırılması sonucu elde edilen gerilim ve akım değerleri



Problem 14.20

Aşağıdaki şekilde verilen devrede A anahtarı t=0 anında kapatıldığına göre Matlab kullanarak tüm akım değişimlerini t ye bağlı bulunuz.

2 adet KGY, 1 adet KAY eşitliği kullanınız $i_1(t=0) = 0, i_2(t=0) = 0, i_3(t=0) = 0$ alınız. Elde edilen eşitliklerden bazıları türev içermiyor ise bu eşitliklerin her iki tarafının bir kez t'ye göre türevini alınız). (L1=2H,L2=3H)



$i[(t), i_2(t), i3(t)$ akımlarını alt alta subplot komutunu kullanarak t;[0 0.1] saniye arasında çizdiriniz.

Çözüm

$$E = L1\frac{di_1(t)}{dt} + L2\frac{di_2(t)}{dt} + i_2(t) * R1 = 10 \qquad (1) \quad \text{KGY}$$

$$i_2(t) * R1 + L2\frac{di_2(t)}{dt} = i3(t) * R2 \qquad (2) \quad \text{KGY}$$

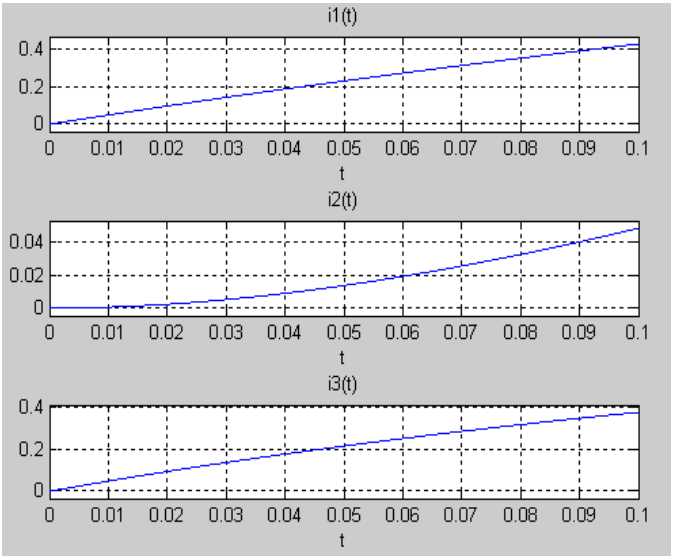
$$i_1(t) - i_2(t) - i_3(t) = 0$$

$$\frac{di_1(t)}{dt} - \frac{di_2(t)}{dt} - \frac{di_3(t)}{dt} = 0$$

$i_1(t) = x; i_2(t) = y; i_2(t) = z$ alınarak(değişken dönüşümü)

Aşağıda verilen problemin çözümünün gerçekleştirildiği Commabd window satırları gösterilmiştir.

```
S=dsolve('2*Dx+3*Dy+4*y=10','4*y+2*Dy-5*z=0','Dx-Dy-
Dz=0','y(0)=0','x(0)=0','z(0)=0')


 S.x                  % x=i1(t) adlı yapının içine girmek için

S.y                       % y=i2(t) adlı yapının içine girmek için

S.z                    % z=i3(t) adlı yapının içine girmek için
subplot(311)
ezplot(S.x,[0,0.1]),grid,title('i1(t)')
subplot(312)
ezplot(S.y,[0,0.1]),grid,title('i2(t)')
subplot(313)ezplot(S.z,[0,0.1]),grid,title('i3(t)')
```


i1(t)
0.4
0.2
0
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
t
i2(t)
0.04
0.02
0
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
t
i3(t)
0.4
0.2
0
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
t

Contents