T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATRİS PROBLEMLERİNE BİLGİSAYAR DESTEKLİ BAZI ÇÖZÜMLER

Levent CİVCİK

YÜKSEK LİSANS TEZİ
MATEMATİK EĞİTİMİ ANABİLİM DALI
Konya, 1998

T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATRİS PROBLEMLERİNE BİLGİSAYAR DESTEKLİ BAZI ÇÖZÜMLER

Levent CİVCİK

YÜKSEK LİSANS TEZİ
MATEMATİK EĞİTİMİ ANABİLİM DALI

KONYA-1998

T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATRİS PROBLEMLERİNE BİLGİSAYAR DESTEKLİ BAZI ÇÖZÜMLER

Levent CİVCİK

YÜKSEK LİSANS TEZİ
MATEMATİK EĞİTİMİ ANABİLİM DALI

Bu tez .....12.....01..... tarihinde aşağıdaki jüri tarafından oybirliği/oy çokluğu ile kabul edilmiştir.

Yrd.Doç.Dr. Cevdet ÇETİN	Prof.Dr. Haydar BULGAK	Doç.Dr. Ömer AKIN
(Danışman)	(Üye)	(Üye)

ÖZET

Yüksek Lisans Tezi

MATRİS PROBLEMLERİNE BİLGİSAYAR DESTEKLİ BAZI ÇÖZÜMLER

Levent CİVCİK

Selçuk Üniversitesi

Fen Bilimleri Enstitüsü

Matematik Eğitimi Anabilim Dalı

Danışman: **Yrd.Doç.Dr.Cevdet ÇETİN**
1998, Sayfa:57

Jüri: **Yrd.Doç.Dr.Cevdet ÇETİN**
Prof.Dr.Haydar BULGAK
Doç.Dr.Ömer AKIN

Bu çalışmada Ax=f lineer cebir sisteminin " Friendly Computer Dialogue System" ile çözülmesi için yazılmıştır. Programın amacı giriş verilerini kolayca bilgisayara girmek ve ekranda kontrol parametrelerini seçerek, sonuçta Ax=f problemini en az hata ile hesaplamak veya bu problemin ill-condition olduğunu tesbit etmektir. Burada kontrol parametreleri arasında, pratik tersleyebilenlerinin parametresi (μ) ve epsilon (ε), çözüme istenilen yaklaşımın derecesidir.

Anahtar Kelimeler: Lineer Cebir Sistemleri , Şart Sayısı, Friendly Computer Dialogue System

ABSTRACT

Masters Thesis

SOME COMPUTER AIDED SOLUTIONS FOR MATRIX PROBLEMS

Levent CİVCİK

Graduate Scholl of Natural and Applied Scienses

Mathematics Education Dept.

Supervisor: **Assist.Prof.Dr.Cevdet ÇETİN**

1998, Pages:57

Jury: **Assist.Prof.Dr.Cevdet ÇETİN**

Prof.Dr.Haydar BULGAK

Assoc.Prof.Dr.Ömer AKIN

One " Friendly Computer Dialogue System" for solving Ax=f problem with bi-diagonal matrix A is introduced in this thesis. The dialog system use a condition number as a measure of sensitivity of the given problem. During computer either detect that given problem is ill-condition or compute the vector in question with guaranteed computer precision.

Key Words: Lineer Algebra Systems , Condition Number, Friendly Computer Dialogue System

TEŞEKKÜR

Yüksek Lisans çalışmalarım süresinde benden hiç bir yardımı esirgemeyen, her zaman gerek bilgisi ve gerekse tecrübesi ile bana daima destek olan Danışman hocam Yrd.Doç.Dr. Cevdet ÇETİN'e ve Prof.Dr. Haydar BULGAK'a, manevi desteklerini esirgemeyen Okul Müdürümüz Doç.Dr. Hüseyin ÖĞÜT'e, bilgisayar konusunda destek olan Öğr.Gör.. Adem GÜNEŞ'e ve kıymetli aileme teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

Sayfa No

1. GİRİŞ

1.1. Lineer Cebir Sistemleri ve Bilgisayarda Hesaplamalar

Teknolojinin hızla ilerlediği, bilgisayar çağı adını verdiğimiz 1990'lı yıllarda, bilgisayarlar günlük hayatımızın bir parçası oldu. Ulaşım ve haberleşmeden, hava raporlarına, tıptan uzaya kadar hemen her yerde bilgisayar kullanımı hızla artmaktadır.

Dünyada, 1945 yıllarında bilgisayar eğitimine başlanmıştır. Türkiye'de de eğitim sisteminde, bilgisayarla eğitime başlama zamanı geldiğine inanıyoruz. Öğrencileri ne kadar erken bilgisayarın son teknik ve modern metotlarıyla tanıştırırsak, eğitim sistemimizde de sonuç o kadar iyi olur. Bu sadece bilgisayar programlama dillerini öğretmek demek değildir.

Bilgisayar; içinde yaşadığımız mühendislik, fen dalları, ekonomide karşılaşılan ve çözümü zor olan yüksek kapasiteli problemlerin hazırlanması ve çözülmesi için büyük fırsat ve kolaylıktır. O halde öğrencilere bu problemleri çözebilecek temel eğitim verilmelidir. Klasik analiz, diferansiyel ve cebir denklemleri, bilgisayar dallarındaki eğitim birbirinden kopuk olarak değil, aynı yaklaşımla ve birbiriyle bağlantılı verilmelidir ki, öğrenci bu problemleri bilgisayarda çözmek için dersler arası bilgi transferi yaparak çözüme ait ideal metodu bulabilsin, rahatlıkla çalışsın, programını yapabilsin ve doğru çözümü bulsun. Bu özellikler yukarıdaki temel eğitimdeki derslerde kullanılan metodun aynı olmasıyla kazanılır.

Buradan anlaşılıyor ki önemli nokta, problemin doğru olarak ortaya konulmasıdır. Matematikte, bilgisayar kullanıcısı, çözeceği problemi anlaması gerekir, yani bu problemin çözümünden ne istediğini bilmesi gerekir. Problemdeki ve çözümdeki neden ve niçin sorularını cevaplayabilmelidir. Şu anda uygulanan eğitim sisteminde sadece problemi çözebilecek metotlar ve yollar, formüller verilerek, bunlar uygulanır denmektedir. Bizim metodumuzda (Garanti yaklaşım metodu)[5] problemi niçin çözdüğümüzü nedenleriyle açıklıyor ve uygulatarak, sonuçları yorumlayarak anlatıyoruz. Bizim metodumuzun dünyadaki diğer son uygulanan metotlardan en büyük farkı ve avantajı ise, fen ve mühendislik dallarında verilen

problemin veriş bilgileri (giriş verileri) ile çözümünün olup olmadığını bularak, çözen kişiyi uyarmaktır. Eğer çözüm yoksa veya giriş elemanlarındaki küçük bir değişime karşı problemin çözümü, büyük tepki verir ise bu probleme ill-condition (kötü konulmuş) problem denir. Aksi halde probleme well-condition (iyi konulmuş) problem denir. Eğer hesap sırasında problemimizin iyi konulduğu tespit edilir ise o zaman problemin bilgisayara göre tam çözümü elde edilir.

Matematik eğitiminin en önemli problemlerinden birisi de A x=f denkleminin çözümüdür. Eğer det A $\neq$ 0 ise keyfi seçilen sıfır olmayan her f için denklemin çözümü var ve tekdir. Biz bu durumu zaten bilmekteyiz. Amacımız bu problemin bilgisayardaki çözümünü incelemek.

Bunu anlayabilmek için bilgisayardaki sayı sistemini gözden geçirmemiz gerekir. Bilgisayarda reel sayılar kümesi yerine FORMAT adlı bir rasyonel alt küme vardır. Yani bilgisayarda γ, P_-, P_+, k parametrelerine bağlı olan

$$F(\gamma, P_-, P_+, K) = \{0\} \cup \{z : z = \pm \gamma^{P(z)}\, m(z),$$
$$m(z) = a_1/\gamma + a_2/\gamma^2 + \ldots + a_K/\gamma^k$$
$$a_1 \neq 0, \quad 0 \leq a_j \leq \gamma-1, \quad j = 1,2,\ldots,k$$
$$P_- \leq P(z) \leq P_+ \}$$

kümeye FORMAT denir. Eğer γ, P_-, P_+' yı sabit olarak görürsek, bu takdirde

$$F_k = F(\gamma, P_-, P_+, k)$$

olarak kabul edebiliriz. Bu kümede

$$\varepsilon_\infty = \gamma^{P+}(1 - \gamma^{1-k}) \quad \text{vardır.}$$

$[-\varepsilon_\infty, +\varepsilon_\infty]$ aralığı vardır. Bu sınırlar her bilgisayarın mikroişlemcisine ve matematik işlemcisine göre değişebilir.

Genel kural, olarak yaklaşımı karakterize eden iki bilgisayar sabiti vardır. Biz bu sabitleri ε_1 ve ε_0 ile gösteririz. ε_1 bilgisayarda 1 sayısına yaklaşım için, ε_0 0

sayısına yaklaşım için gösterilen sabitlerdir. Bilgisayarda $(0,\varepsilon_0)$ ve $(1,\ 1+\ \varepsilon_1)$ aralığında bir sayı yoktur.

Bilgisayarda z reel sayısını, hafızaya yerleştirirken üç durum ile karşılaşabiliriz;

1. z, seçilen Formatın elemanı olduğunda hiç hatasız hafızaya yerleşir,

2. z yerine fl(z), seçilen Formatın z'e en yakın sayısı olarak yerleşir ve bu durumda

$$| z - fl(z) | \leq \varepsilon_0$$

veya

$$| z - fl(z) | \leq \varepsilon_1 | z |$$

hata ortaya çıkar, bunu da toplam

$$| z - fl(z) | \leq \varepsilon_1 | z | + \varepsilon_0$$

şeklnde gösterebiliriz.

3. z, seçilen Formatın dışında ise, bu durumda bilgisayar OVERFLOW hatasını gösterir ve kilitlenir.

Aynı şekilde, A NxN tipinde reel bir matris ve f Nx1 tipindeki reel vektör bilgisayarda hafızaya yerleşirken, OVERFLOW durum ile karşılaşmazsa ,A ve f yeterince büyük ise, yani $\| A \| > \varepsilon_0$ ve $\| f \| > \varepsilon_0$ A yerine fl(A) ve f yerine fl(f) şartıyla

$$\| A - fl(A) \| \leq \varepsilon_1 \| A \|$$

$$\| f - fl(f) \| \leq \varepsilon_1 \| f \|$$

şartlarını sağlar. Buna giriş hataları denir. Burada $\| A \|$ A matrisinin spektral normunu ve $\| f \|$ f vektörünün Öklid normunu kullanacağız. Yani

$$\| f \| = \sqrt{\sum_{i=1}^{N} f_i^2} \ ; \ \ \| A \| = \max_{\|x\|=1} \| Ax \|$$

dir.

Basit bir örnek ele alalım. $3x = 1$. Bilgisayarda hafızaya hiç hatasız 3 ve 1 yerleşiyor. Bu problemin çözümü de $x = 0.33333...$ olur ve bu reel sayı formatının dışında olduğundan verilen probleminin tam çözümü bulmak imkansızdır. Böylece çözümün ne olduğunu önceden açıklamak mecburiyetimiz vardır.

Farz edelim ki klasik manada $x = 1/3$ çözümünü bulduk. O zaman bilgisayarın formatından dolayı giriş hatası ile karşılaşırız. Yani bilgisayarda elde edilen çözüm $fl(x)$ ile gerçek çözüm ;

$$\| x - fl(x) \| \leq \varepsilon_1 \| x \|$$

eşitsizliği ile bağlıdır. Buradan sonuç olarak; $Ax = f$ problemi bilgisayarda hesap edilmek istenirse, önce eps parametresi kullanıcı tarafından bilinmeli ve çözüm olarak $\| x - y \| \leq eps \| y \|$ eşitsizliğini sağlayan y vektörünü kullanma mecburiyeti vardır. Bu parametre büyük olabilir fakat hiç bir zaman ε_1'den küçük olamaz. Eğer kullanıcı daha az rölatif hata ile probleminin çözümünü bulmak isterse, önceden seçilen formatdan daha kuvvetli bir Format seçmesi gerekir.

Dolayısıyla, eps parametresinin ne olduğunu ve niçin verilen $Ax = f$ problemi için önemli olduğunu anlamış olduk.

Şimdi bizim ana problemimize geçelim. Yani bilgisayarda iki köşegenli lineer denklem sistemini hesaplamak için ne yapmalıyız ?

$$\begin{aligned} a_1x_1+b_2x_2 &= f_1 \\ +a_2x_2+b_3x_3 &= f_2 \\ &\cdots\cdots\cdots \\ a_{N-1}x_{N-1}+b_Nx_N &= f_{N-1} \\ a_N x_N &= f_N \end{aligned}$$

sistemini ele alalım. Burada a_i , b_j ve f_i verilen reel sayılar , x_i istenilen vektörünün elemanlarıdır ($i = 1,2, \ldots , N$, $j = 2, 3, \ldots , N$).

Yukarıdaki sistemi bilgisayarda hesaplamak mümkün mü? Bunu incelemeye çalışalım.

Açıktır ki , eğer $a_i \neq 0$ ise (i = 1, 2, ... ,N) verilen problemin çözümü vardır ve tekdir. Bu çözüm aşağıdaki gibidir.

$$x_N = f_N / a_N$$
$$x_{N-1} = (f_{N-1} - b_N x_N) / a_{N-1}$$
.....
$$x_1 = (f_1 - b_2 x_2) / a_1 .$$

Bu şart gerek ve yeter şarttır. Bunu bilgisayarda uygulayabilir miyiz ? Bir örnek alalım.

$$x_1 - 2x_2 = 0$$
$$x_2 - 2x_3 = 0$$
.....
$$x_{N-1} - 2x_N = 0$$
$$x_N = 1$$

Burada, bütün köşegen elemanları 1'dir. Yani önceki kuralına göre verilen probleminin çözümü var ve tektir. Bu çözüm

$$x_j = 2^{N-j}, \quad j = 1, 2, \ldots , N$$

dir.

Fakat bu bilgisayar açısından tehlikeli bir çözüm olabilir. Çünkü bilgisayar sayılarının kümesi sınırlıdır, yani ε_∞ dan daha büyük bilgisayar sayısı yoktur.

Dolayısıyla herhangi bir ε_∞ sayı için $2^{N-1} > \varepsilon_\infty$ olmak üzere $N = N(\varepsilon_\infty)$ sayısı vardır. Bu şekilde seçilen N için verilen problemi bilgisayarda hesaplama sırasında OVERFLOW durum ile karşı karşıya geliyoruz.

Klasik matematiğin eğitiminde iyi bilinen Kramer kuralı Ax = f deklemi için kullanıldığında det A değerine dayanıyor. Bilgisayarda bu kuralın başarısız olduğunu biraz önce görmüştük. Ayrıca det A'nın başarısız olduğunu gösteren bir örnek daha verelim.

$$1/2x_1=1$$
$$1/2x_2=1$$
$$\cdots\cdots\cdots\cdots$$
$$1/2x_N=1$$

sistemini ele alalım.

Bu örnekte $a_j = 1/2$, $j = 1, 2, \ldots, N$ olduğundan çözüm var ve tekdir.

Bu sistemin katsayı matrisi A diagonal bir matristir. Açıktır ki

$$\det A = 1/2^N$$

ele alalım.

Dolayısıyla herhangi bir ε_0 sayı için $1/2^N < \varepsilon_0$ olmak üzere $N = N(\varepsilon_0)$ sayısı vardır. Yani bilgisayar $1/2^N$'ni sıfır kabul eder. Buna göre bilgisayar det A=0 olarak gösterir. Böylece çok iyi konulmuş problemin çözümünün olmaması sonuç olarak verilir.

Bu örneklerden, klasik matematikle verilen KRAMER kuralı gibi yöntemlerin bilgisayarla çözümlerde başarısızlığa yol açabileceğini söyleyebiliriz.

Bilgisayar destekli matematik eğitimi gibi bir alanda, sağlam temellere oturmayan yöntemlerle işe başlanırsa ilerde oldukça önemli problemlerin doğacağı aşikardır. Bu durum son 50 sene içinde net şekilde belirlenmiştir. Bu konuda John von Neumann [4] ve Turing [3] tarafından çalışmalar başlatılmıştır. Sonra örnek olarak [1],[2] çalışmalar ile verilen problemi için $\mu(A) = \| A \| \| A^{-1} \|$ şart sayısı olarak belirlenmiş ve $\mu(A)$'ya göre problemler ele alınarak değerlendirilmiştir.

Bu şart sayısının iki önemli özelliği vardır. Onları aşağıdaki iki teoremle gösterebiliriz.

Teorem 1.1.1. Eğer $\mu(A) < \infty$ ve $2\mu(A)\| B \| / \| A \| < 1$ ise bu taktirde

$\mu(A+B) < \infty$ ve $| \mu(A) - \mu(A+B) | \leq 3 \mu^2(A)\| B \| / \| A \|$

olur.

Teorem 1.1.2. Eğer $\mu(A) < \infty$ ve $2\mu(A)\| B \| / \| A \| < 1$ ise bu taktirde $Ax = f$ ve ona yakın olan $(A+B)y = f+g$ çözümlerinin birbirine yakınlığı

$$\| x - y \| / \| x \| \leq 3\, \mu(A) \, (\| B \| / \| A \| + \| g \| / \| f \|)$$

eşitsizliği ile verilir [1].

Bu teoremlerin ispatları [1] bulunabilir.

Buradan açıktır ki, $\mu(A)$ ne kadar büyük ise verilen $Ax=f$ denkleminin çözümü A ve f'deki değişikliklere hassastır. Yani $\mu(A)$ ne kadar küçük ise verilen $Ax = f$ problemi o kadar iyi konulmuştur.

Uygulamalarda verilerin ne kadar tam olup olmadığı iki değişik yoldan kaynaklanabilir. 1) Aproksimasyon hatalar ile ölçü hataları 2) Verilerin, bilgisayarın hafızasına yerleşmesi zamanında ortaya çıkan yuvarlama hatalarıdır. Bundan dolayı doğal olarak $Ax = f$ denklemi yerine $(A+B)\, y = f + g$ denkleminin ona "yakın" bir denklem kabul edebiliriz. Biz sadece $\| B \| / \| A \| < eps$ ve $\| g \| / \| f \| < eps$ olduğunu biliyoruz. Burada eps parametresi bize giriş datalarının ne kadar iyi verilip verilmediğini gösteriyor. Bu açıdan eps parametresi çok önemlidir .Eğer hatalar sadece bilgisayara yerleşim hataları ise bu taktirde $eps = \varepsilon_1$ olur

$Ax = f$ problemini hesap etmek istiyorsak, çok fazla hassas sistemler bizim pratikte işimize yaramaz. Çünkü hesaplanan çözüm, gerçek problemin çözümüne ne kadar yakın olup olmadığı hakkında yorum yapamıyoruz. Bunda bir çıkış yolu olarak: μ^* pratik tersleyebilen matrislerinin üst sınırı olarak kabul edilen bir parametreyi kullanmak olabilir (bak [5],[15]). Bu parametre bize teorem 1'i değişik şekilde yazmaya imkan tanıyor.

Teorem 1.1.3. Eğer $\mu(A) < \mu^*$ ve $2\mu^* \| B \| / \| A \| < 1$ ise bu taktirde

$$\mu(A+B) < 1.5\mu^* \quad \text{ve}$$

$$| \mu(A) - \mu(A+B) | \leq 3\, \mu^2(A) \| B \| / \| A \|$$

olur.

Bu teorem μ^* değeri ile giriş elemanlarının ne kadar doğru verildiği ile ilgilidir. Bu teoreme göre $\mu^* < \| A \| / (2\| B \|)$ olarak seçilmelidir.

Eğer $20\mu(A)\varepsilon_1 < 1$ ise $Ax = f$ problemi için $\| y - x \| < \varepsilon_1 \|x\|$ bilgisayara göre "tam" çözümü bulabiliriz. Böylece bilgisayar açısından doğal şekilde $\mu^* = 1/(20\varepsilon_1)$ olarak tanımlanabilir.

Uyarı: Kullanıcı bilgisayarda iki köşegenli sistemleri hesaplamak için iki kontrol parametresi seçmeye mecburdur. Bunlar eps ve μ^* dır. Fakat $eps > \varepsilon_1$ ve $\mu^* > 1/(20\varepsilon_1)$ olmalıdır. Ve bu iki parametre standart formatın $\mu^* = 1/(20\varepsilon_1)$ değer ile sınırlıdır. Eğer bu sınırlamanın dışındaki bir problem mutlaka hesap edilmeli ise, bu taktirde Standart Format yeterince geniş seçilmelidir.

Verilen problemin nasıl bir problem olduğunu, yani iyi yada kötü olup olmadığını bilmek önemlidir. Bunun için problemin türüne göre çeşitli kriterler vardır. Mesela; matematikte çok iyi bilinen klasik matris denklemi $Ax = f$ problemi $\det A \neq 0$ olduğunda sıfır olmayan her f için bir çözüme sahiptir. Bu çözüm var ve tektir. Bu problem iyi bir problemdir. Eğer $\det A = 0$ ise problemi çözmek için bazı yollar vardır. Ancak çözüm ya bulunamayacaktır ya da tek olmayacaktır. Yani bu problem bizim için kötüdür.

$$Ax = f \quad , \quad \det A \neq 0$$

probleminin f `e bağlı olmaksızın (her f için) çözümü vardır ve tektir. Bu problem iyi tanımlı bir problemdir. Ancak A`nın elemanları değiştirilmeye başlandığında bu problemin durumu değişecektir. Yani iyi tanımlı olmayabilecektir. Mesela;

$$A=\begin{pmatrix} \varepsilon & 0 \\ 0 & 1 \end{pmatrix}$$

olarak aldığımızda $\det A = \varepsilon \neq 0$ olup her f için çözüm vardır ve tektir. Buna karşılık

$$B=\begin{pmatrix} -\varepsilon & 0 \\ 0 & 0 \end{pmatrix}, \qquad \| B \| = \varepsilon$$

matrisi için (A + B) y = f probleminin durumu, det (A + B) = 0 olduğundan bazı f'ler için sonsuz çözüm varken bazı f'ler için ise çözüm yoktur. Yani; (A+B) y = f problemi iyi tanımlı değildir. Çünkü çözümün varlığı ve tekliği f'nin elemanlarına göre değişkenlik arz etmektedir.

Bu tür problemlerde akla şu soru gelebilir: Problem iyi tanımlı olmayan bölgeye ne kadar uzak (veya yakın)dır? Acaba A'nın elemanlarını ne kadar değiştirirsek problem yine iyi tanımlı olacaktır?

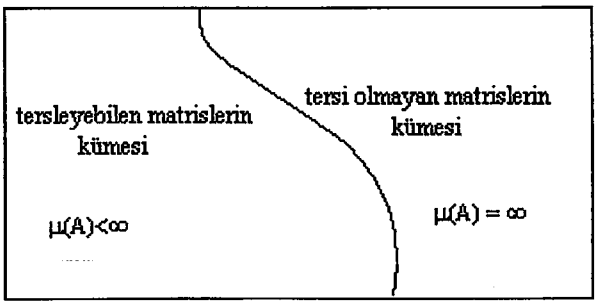
Yukarıda ki (A + B) y = f problemi için sorunun cevabı; Ax=f iyi tanımlı ise ($\|B\| / \|A\|$) < (1 / 10 $\mu(A)$) olacak şekildeki her B-küçük (normu küçük) matris için (A + B) y = f problemi de iyi tanımlıdır. Burada $\mu(A) = \| A \| \| A^{-1} \| \geq 1$, Ax = f probleminin şart sayısıdır. Buna göre,

$$A=\begin{pmatrix} \varepsilon & 0 \\ 0 & 1 \end{pmatrix}$$

için Ax = f problemi iyi tanımlı iken (A + B) y = f probleminin de iyi tanımlı olması için B matrisi, $\mu(A) = 1 / \varepsilon$, $\| A \| = 1$ olduğundan $\| B \| < \varepsilon / 10$ olacak şekilde küçük olmalıdır. Başka bir ifade ile $\| B \| < \varepsilon / 10$ olan bütün B matrisleri için (A + B) y=f problemi iyi tanımlıdır.

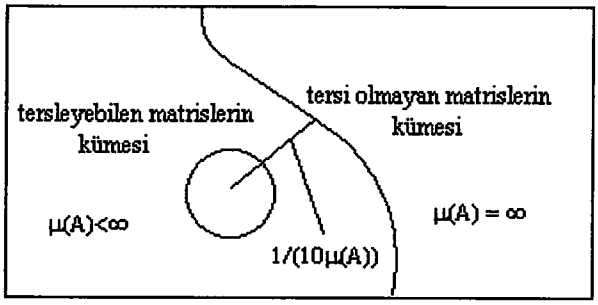
Çözümü var, tek ve elemanlarına göre sürekli olan bu tür problemlere iyi konulmuş (korekt) problem denir.

Matrislerin kümesini $\mu(A)$ şart sayısı ile iyi konulmuş ve iyi konulmamış diye iki parçaya bölmek mümkündür. Bunu bir şekille gösterelim.



Şekil 1.1.1.

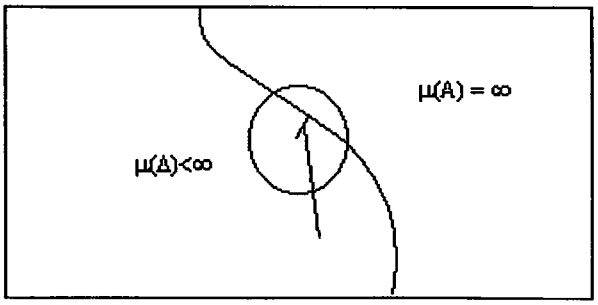
$\mu(A) < \infty$ ise $A\,x = f$ problemi iyi konulmuş bir problemdir. Bir A matrisinin iyi konulmamış matrislerden uzaklığının $1/[10\mu(A)]$ olduğunu söylemek mümkündür. Bunu bir şekil ile gösterebiliriz.



Şekil 1.1.2.

Bir matrisin (veya problemin) iyi konulmuş matrislere (yada problemlere) olan uzaklığı ile ilgili yapılmış çok çalışma vardır [1], [2], [7], [9].

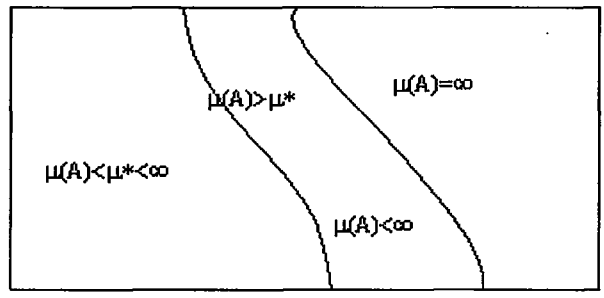
Bu kaynakların bir çoğundaki problemlerde "belirsizlik prensibi" vardır. Yani bir A matrisini $\{A, \|A-D\| < \varepsilon \|D\|,$ D verilen matris , $\mu(D) < \infty\}$ kümesinden aldığımızda, eğer $\mu(D)\varepsilon > 1$ doğru olursa biz A matrisinin iyi konulmuş veya iyi konulmamış matris olup olmadığını söyleyemeyiz. Çünkü D matrisi kendisi iyi konulmuş olmasına rağmen iyi konulmamış matrislerin kümesine çok yakındır. Bunu başka bir şekille izah edelim.



Şekil 1.1.3.

Bir A matrisinin iyi konulmamış matrislerin kümesine olan uzaklığını öğrenebilmek için pratik iyi konulmuş problemleri tanıtmak gerekmektedir.

Ax = f problemi, eğer $\mu(A) = \| A \| \| A^{-1} \| < \mu^* < \infty$ olacak şekilde bir μ^* sayısı verilebilirse o zaman pratik iyi konulmuş problemdir. Aksi takdirde pratik iyi konulmamış problemdir. Burada μ^*, pratik iyi konulmuş ve pratik iyi konulmamış matrislerin arasında bir sınır oluşturan sayıdır. Bunu bir şekille gösterelim.



Şekil 1.1.4.

O halde sonuç olarak bizim metodumuzun önceki satırlarda da belirttiğimiz gibi en büyük avantajı problemin (sistemin) $\mu(A)$ şart sayısını verilen formülle çok net şekilde hesaplayarak, problemin ill-condition veya well-condition olup olmadığı hakkında kullanıcıyı uyarmak, problemin ill-condition yapan data ve giriş bilgilerinin veya metodun bir kısmının veya tamamının değiştirmesi gerektiğini göstererek, zaman ve emek tasarrufu sağlamaktır.

1.2. Friendly Diyalog Sistemleri

XSC (eXtention for Scientific Computations) [11], MAPPLE [12] gibi bilgisayar için sistemler elemanter işlemlerin küçük hata ile hesaplanmasını sağlıyorlar. Ayrıca friendly computer sistemi, diyalog bir sistemdir. Onun diyaloğu ideal olarak görülmektedir. Ayrıca Matlab da iyi bilinen lineer cebir problemlerini hesap eden bir diyalog sistemdir. Bu sistem MAPPLE'ın diyalog sistemine göre kullanıcıyı daha dikkatli olmaya mecbur kılıyor. Bu sistemler kullanıcıya istenilen Ax= f problemini hesap etme imkanı verir. S.Ü. Uygulamalı Matematik Araştırma Merkezinde friendly diyalog sistemlerin üzerinde çalışmalar yapılmaktadır. Borland Pascal'a dayanarak Oğuzer Sinan tarafından bir matrisin spektral portresini hesaplamak için bir friendly diyalog sistem ortaya konulmuştur[14]. Bu sisteme dayanarak Oğuzer Sinan ile birlikte Necati Taşkara kendi doktora tezinde benzer şekilde periyodik sistemlerin kararlığını incelemek için bir sistem kurmuştur. Bu çalışmalar devam ettirilmiştir. Delphi programlama dili kullanarak [10] iki köşegenli bir A matrisi için, Ax= f lineer cebir sistemini hesaplamak için bir friendly diyalog sistem yapılmıştır.

1.3. Tezin Yapısı

Bu yüksek lisans tezi 4 bölümden ibarettir. Bölüm 1'de tezin amaçları ve tarihten bazı açıklamalar yapılmıştır. Bölüm 2'de konu ile ilgili gerekli matematik temel kavramlar verilmiştir ve onların için bazı teoremler de hatırlatılmıştır. Bölüm 3'te Delphi programlama dili hakkında bilgiler yer almaktadır. Bölüm 4'te ise geliştirmeye çalıştığımız diyalog sistemi hakkında ayrıntılı bilgiler bulunmaktadır.

2. PROBLEMİN MATEMATİK TARAFI

2.1. Singüler Değerler

Tanım 2.1.1. A, Σ, U, V NxN tipinde reel , U, V ortogonal, Σ köşegen matrisler, onun köşegen elemanları $0 \le \sigma_1(A) \le \sigma_2(A) \le \dots \le \sigma_N(A)$ sıralanmış negatif olmayan sayılar olmak üzere

$$A = U\Sigma V,\ \Sigma = \begin{pmatrix} \sigma_1(A) & & & \\ & \sigma_2(A) & & \mathbf{O} \\ & & \ddots & \\ & \mathbf{O} & & \sigma_{N-1}(A) \\ & & & & \sigma_N(A) \end{pmatrix}$$

ifadesini sağlıyor ise bu taktirde $\sigma_1(A), \sigma_2(A), \dots, \sigma_N(A)$ var ve tektir. Bu sayılara A matrisinin singüler değerleri denir (bakınız [1]).

Eğer $\sigma_1(A)>0$ ise o zaman A matrisinin tersi A^{-1} var ve

$$A^{-1} = V^*\Sigma^{-1}U^*,\ \Sigma^{-1} = \begin{pmatrix} 1/\sigma_1(A) & & & \\ & 1/\sigma_2(A) & & \mathbf{O} \\ & & \ddots & \\ & \mathbf{O} & & 1/\sigma_{N-1}(A) \\ & & & & 1/\sigma_N(A) \end{pmatrix}$$

dir.

A matrisinin spektral normunu $\| A \|$ ve x vektörünün Öklid normu $\| x \|$ ifadelerini kullanarak aşağıdaki ifadelerin doğru oldukları ispat edilebilir.

1. $\| A \| = \max_{\|x\|=1} \| Ax \| = \sigma_N(A)$;

2. Eğer $\sigma_1(A) > 0$ ise o zaman $\| A^{-1} \| = 1/\sigma_1(A)$;

3. Eğer $\lambda_1(A^*A) \le \lambda_2(A^*A) \le \dots \le \lambda_N(A^*A)$ sıralanmış A^*A matrisinin özdeğerleri ise o zaman $\sigma_j(A) = \sqrt{\lambda_j(A^*A)}$, j = 1,2, ... , N olur.

4. Eğer

$$H = \begin{pmatrix} 0 & \mathbf{A} \\ \mathbf{A}^* & 0 \end{pmatrix}$$

$\lambda_1(H) \le \lambda_2(H) \le ... \le \lambda_N(H)$ sıralanmış H matrisinin özdeğerleri ise o zaman

$$\sigma_j(A) = \lambda_{N+j}(H) = -\lambda_{N+1-j}(H),\ j = 1,2, ... , N$$

olur;

5. $\sigma_N(A) - \| B \| \le \sigma_N(A+B) \le \sigma_N(A) + \| B \|$;

6. $\sigma_1(A) - \| B \| \le \sigma_1(A+B) \le \sigma_1(A) + \| B \|$.

2.2. Şart Sayısı (Condition Number)

Şart sayısı $\mu(A) = \| A \| \| A^{-1} \|$ olarak seçilir ise o zaman aşağıdaki özellikler vardır.

1. Eğer A^{-1} yoksa $\mu(A) = \infty$ kabul edilir.
2. $\mu(A) = \sigma_N(A)/\sigma_1(A)$
3. Burada $\sigma_j(A)$ A matrisinin sıralanmış singüler değeri

$$\sigma_1(A) \le \sigma_2(A) \le ... \le \sigma_N(A),$$

$$\sigma_j(A) = \sqrt{\lambda_j(A^* A)},\ j = 1, 2, ... , N$$

$\lambda_j(A^*A)$ A matrisinin öz değerleridir.

4. $\alpha \neq 0$, keyfi seçilen bir sayı olmak üzere $\mu(\alpha A) = \mu(A)$ dır.
5. Q, V ortogonal matrisler olmak üzere $\mu(QAV) = \mu(A)$ dır.
6. $\mu(A) \ge 1$ dir.

Ax = f probleminin çözümü sırasında elde edilen bir y vektörünün, istenilen x vektörüne ne kadar yakın olup olmadığını neye dayanarak söyleyebiliriz? Önce bir tanım verelim:

Tanım 2.2.1. $\| x-y \|$ değerine *mutlak hata* denir.(yani y istenilen vektörün, x vektörüne ne kadar yakın olduğunu gösterir) Fakat bazen bu değer de küçük olabilir.

Fakat x küçük olduğundan bu değer x hakkında yeterince bilgi vermiyor. Bunun için $\| x-y \| / \| x \|$ *bağımlı hata* da çok önemlidir.

Şart sayısının tanımından açıktır ki

$$Ax = f, \qquad A(x + \xi) = f + g$$

bir birine yakın sistemler için

$$\| \xi \| \leq \|A^{-1}\| \, \|g\|, \quad \frac{\|\xi\|}{\|x\|} \leq \mu(A) \frac{\|g\|}{\|f\|}$$

doğrudur.

Bu eşitsizlikler, çözümünü hesapladığımız y vektörüne ait mutlak ve bağımlı hatayı gösterir. Bağımlı rezidü şart sayısıyla çarpıldığında daha büyük olabilir.

Şüphesiz o bağımlı rezidü şart sayısına bölündüğünde daha küçük olabilir. Böylece şart sayısının çok büyük olduğu zaman rezidü vektörü y'nin yaklaşımı hakkında çok az bilgi verir. Şart sayısı 1'e yakın olduğu zaman rezidü y'nin bağımlı hatasının iyi bir ölçümüdür.

Örnek 2.2.1. $Ax = f, \qquad A = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix}; \quad f = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$

Bu durumda $x = (0, 1/10)^T$. Eğer $g = (g_1, g_2)^T = (\varepsilon, 0)^T$ ise $\frac{\|g\|}{\|f\|} = \varepsilon.$

$A\xi = g$ denkleminden ξ'yi bulduğumuz zaman $\xi = (\xi_1, \xi_2)^T = (\varepsilon, 0)^T$ olur.

Açıktır ki,

$$\|\xi\| = \varepsilon; \quad \frac{\|\xi\|}{\|x\|} = 10\varepsilon = \mu(A) \frac{\|g\|}{\|f\|}.$$

$$\frac{\|\xi\|}{\|x\|} \leq \mu(A) \frac{\|g\|}{\|f\|}$$

LEMMA 2.2.1. *Eğer* N×N *boyutlu* A *matrisinin tersi alınabiliyorsa*

$$(\mu(A) < \infty) \text{ ve } \mu(A)\frac{\|B\|}{\|A\|} < 1 \text{ ise}$$

A+B *de tersi alınabilen matrisdir ve*

$$\| (A+B)^{-1}\| \leq \| A^{-1}\| \, (1-\mu(A)\frac{\|B\|}{\|A\|})^{-1}$$

sağlar.

Bu lemma bize tersleyebilen matrislerin kümesinin, tersi olmayan matrislerin kümesine olan uzaklığını garantili olarak gösteriyor. Bunu daha net olarak anlamak için aşağıdaki şekli verelim.

Şekil 2.2.1.

Lemma 2.2.1. i kullanarak süreklilik teoremi adını verdiğimiz teoremi ispatlayabiliriz (bakınız [1]).

TEOREM 2.2.1. *Eğer* N×N *boyutlu* A *matrisi tersi alınabilir bir matris ise* $(\mu(A) < \infty)$ *ve*

$$4\mu(A)\frac{\|B\|}{\|A\|} < 1$$

ise bu taktirde A+B *de tersi alınabilir matristir ve*

$$|\mu(A) - \mu(A+B)| \leq 3\mu^2(A)\frac{\|B\|}{\|A\|}$$

dir.

Kötü konulmuş (ill-condition) matrise örnek.:

$$A = \begin{pmatrix} 1 & 2 & & & \\ & 1 & 2 & \mathbf{O} & \\ & & \ddots & \ddots & \\ & \mathbf{O} & & 1 & 2 \\ & & & & 1 \end{pmatrix}$$

NxN tipinde bir matris olsun. Açıktır ki, $\|A\| < 3$.

$$f = (0, 0, \dots, 0, 1)^T; \quad A^{-1} f = ((-2)^{N-1}, \dots, 2^2, -2, 1)^T$$

den ve

$$\frac{\|A^{-1}\|^2}{\|f\|^2} = 1+2^2 + 2^4 + \dots + 2^{2(N-1)} > 4^{N-1}$$

sonucunu çıkarabiliriz.

$$\sigma_1(A) = \max_{\xi \neq 0} \frac{\|\xi\|}{\|A\xi\|} = \min_{f \neq 0} \frac{\|f\|}{\|A^{-1}f\|} < 2^{1-N}.$$

Buradan $\mu(A) > 3 \times 2^{N-1}$ ve detA = 1. Yani N ne kadar büyük ise A matrisinin şart sayısı da büyür. Fakat det A = 1 ve N boyut ile hiç bağlı değildir. Böylece de detA ya göre yorum yapmak oldukça sağlıksızdır. Ayrıca det A fazla hassastır. Kolayca görülür ki,

$$A_\varepsilon = \begin{pmatrix} 1 & 2 & & & \\ 0 & 1 & 2 & \mathbf{O} & \\ \vdots & & \ddots & \ddots & \\ 0 & \mathbf{O} & & 1 & 2 \\ \varepsilon & 0 & \cdots & 0 & 1 \end{pmatrix}$$

matrisi A ya çok yakındır $\| A - A_\varepsilon \| = \varepsilon$ ve $|(\det(A)-\det(A_\varepsilon)/ \det(A)| = 1$, $\varepsilon = 2^{1-N}$ olması durumunda (eğer N=2m ise) öyle ki det A_ε = 0'dır. Aşağıdaki teorem bize determinantın değerinin A matrisindeki küçük değişikle ne kadar bağlı olduğunu gösteriyor.

TEOREM 2.2.2. *Eğer* N×N *boyutlu* A *matrisinin tersi alınabilir*

$$(\mu(A) < \infty) \text{ ve } N\,\mu(A)\frac{\|B\|}{\|A\|} < 1 \text{ ise}$$

$$|(\det(A) - \det(A+B))/\det(A)| \le N\,\mu(A)\frac{\|B\|}{\|A\|}(1 - N\mu(A)\frac{\|B\|}{\|A\|})^{-1}$$

sağlanır.

Burada bu teoremin verilmesindeki sebep şudur: bildiğimiz gibi matematik eğitiminde determinant kavramı oldukça önemlidir. Bu teoremle determinanttaki hassasiyeti $\mu(A)$'ya bağlı olarak güzel bir şekilde ifade etmekteyiz.

2.3. Verideki (Datadaki) Belirsizlikler

TEOREM 2.3.1. (*Datadaki belirsizlikler*). *Eğer* A, N×N *boyutlu tersi alabilen* $(\mu(A)<\infty)$ *matrisi ise ve* $Ax = f$ *sistemiyle beraber ona çok yakın* $(A+B)y=f+g$ *sistemimiz varsa öyle ki;*

$$\frac{\|B\|}{\|A\|} \le \beta, \frac{\|g\|}{\|f\|} \le \alpha,$$

burada $\mu(A)\,\beta< 1$ *ise*

$$\frac{\|x-y\|}{\|x\|} \le \frac{\mu(A)(\alpha+\beta)}{1-\mu(A)\beta}$$

eşitsizliği doğrudur.

2.4. Rezidü Problemi

TEOREM 2.4.1. *Eğer* A, N×N *boyutlu tersi alabilen* $(\mu(A) < \infty)$ *matrisi ise o zaman* $Ax = f$ *sisteminin tek çözümü x ve herhangi bir y vektörü aşağıdaki eşitsizliği*

$$\frac{\|x-y\|}{\|x\|} \le \mu(A)\frac{\|Ay-f\|}{\|f\|}$$

sağlarlar.

2.5. Pratik Tersi Alınabilen Matrisler

Bir A matrisinin pratik tersleyebilmesinin parametresi μ^* ile verilir.

Eğer A tersleyebilen bir matris ve $\|B\| / \|A\| < 1/\mu^*$ olmak üzere tüm A + B matrisler de tersleyebilen ise o zaman A matrisine pratik tersleyebilen matris(μ^*-tersleyebilen) denir. A'nın tersleyebilmesinin parametresi $\mu(A)$ olmak üzere, $\mu(A)<\mu^*$ eşitsizliği A matrisinin pratik tersleyebilmesini garanti edeceği açıktır.

Kullanıcıların, μ^* değerini belirli fiziksel problemlere bağlı olarak seçmeleri tabiidir. Çünkü A matrisinin elemanlarını tam olarak genellikle bilmiyoruz . Onların bazı belirsizliklerini B dejenere matrisi ile veriyoruz. Burada $\| B \| / \| A \| < eps$ olup, eps verilen küçük pozitif bir sayıdır. Bundan dolayı μ^*'ı, $\mu^* = 2/eps$ olarak seçebiliriz.

Bu durumda ispatlanan hassasiyet teoremi bize μ^*'ın , tersleyebilen A matrisine yakın bütün A + B matrisleri için, ($\|B\| / \|A\| < 1/(2\mu^*)$) $1.1\mu^*$'ın tersleyebilen olduğu sonucunu çıkarmamıza imkan tanır.

μ^* 'ın seçilmesi için diğer bir yol da , $\mu(A)$ 'nın hesaplanmasına ve kullanılan özel bilgisayara bağlı olarak seçilmesi yoludur. Bilgisayarı karakterize eden ε_1 bir parametre (1 ve $1+ \varepsilon_1$ sayılar arasında hiç bir bilgisayar sayısı yoktur) olmak üzere ve 20 $\mu(A)\ \varepsilon_1 < 1$ doğru ise o zaman $Ax = f$ sisteminin tek çözümü için bu bilgisayarla yaklaşık çözüm hesaplanabilir ve

$$\| x - y \| / \| x \| < \mu(A)\ \varepsilon_1$$

olur (bakınız [1]). Dolayısıyla $\mu^*=1/(20\varepsilon_1)$ olarak seçilebilir.

μ^* değerinin tanıtılmasını $\mu(A)$'nın hesaplanma yöntemine bağlı olarak düzenledik. Bu yöntem ya μ^*'ın tüm doğru değerlerinin hesaplanmasına (onun bilgisayar gösterimindeki son değeri hariç olmak üzere), ya da A tersleyebilmesinin, matrisin kötü şartlarının bulunmasına bağlıdır. Bu ise $\mu(A) > \mu^*$ eşitsizliği ile verilir.

2.6. Format

Bu bölüm Lineer Cebir Sistemlerinin Şart Sayısı [8] seminerine dayanılarak yazılmıştır. Bilgisayarda hesaplamaların algoritmalar ile yapılmasını anlatmadan önce bilgisayarın karşımıza hangi ek şartları getirdiğini hatırlatmamız gerekir. Bu şartların en başında bilgisayarın bildiğimiz reel sayılar yerine onun bir alt kümesi ile işlem yaptığı yer alır. Reel sayılar özel bir format ile bilgisayar sayıları olarak yazılabilir. Bu formatı oluştururken bazı giriş hataları meydana gelebilir. Demek ki bu giriş hatalarını göz önüne almalıyız. Diğer yandan hesap sürecinde karşımıza çıkan yuvarlama hatalarını kontrol altına almalıyız. Bu hataları kontrol eden algoritmalara ve programlara *Garanti Yaklaşım Algoritmaları ve Programları* denir. Bu programlarda bilgisayarda hesaplanan problem veya bu problemin çözümü giriş hatalarından daha fazla hata taşımıyor. Ayrıca bu programlar, problem iyi konulmamış ise tespit ediyor ve kullanıcıya bildiriyor.

2.6.1. Format'ın Tanımı

Q rasyonel sayılar kümesinin bir alt kümesi olan γ, P_+ ve k doğal sayılar, P_- negatif tam sayı olmak üzere

$\mathbf{F}= \mathbf{F}(\gamma, P_-, P_+, k) = \{0\}\cup\{z: z = \pm\gamma^p (m_1\gamma^{-1} + m_2 \gamma^{-2} + ... + m_k \gamma^{-k}),$ $P_- \le p \le P_+; m_1, m_2, ... , m_k$ tam sayılar 0 ve γ -1 arasında, $m_1 \neq 0\}$. şeklinde tanımlı $\mathbf{F} = \mathbf{F}(\gamma, P_-, P_+, k)$ kümesine **Format** denir.

1) Her bir Format'ın ε_∞ en büyük ve $\varepsilon_{-\infty}$ en küçük elemanları vardır. Bu elemanlar γ, P_-, P_+, ve k parametrelerine bağlıdır.

2) ε_0 sayısına **F** kümesinin *en küçük pozitif elemanı* denir. Açıktır ki;

$$\varepsilon_0 = \gamma^{P_-} \times(1\times\gamma^{-1} + 0\times\gamma^{-2} + ... + 0\times\gamma^{-k})_{min} = \gamma^{P_- -1}$$

dir.

F kümesinde 0 ile ε_0 arasında **F**'nin başka bir elemanı yoktur. Diğer bir ifadeyle ($-\varepsilon_0, \varepsilon_0$) aralığında **F**'nin elemanı olarak yalnızca 0 (sıfır) bulunur.

3) $\varepsilon_1 = \gamma^{1-k}$ verilen **F** kümesininin önemli bir karakteristiğidir. $(1, 1 + \varepsilon_1]$ aralığında **F** kümesinin tek bir elemanı vardır ve bu da $(1 + \varepsilon_1)$ dir.

Demek oluyor ki;

$\varepsilon_\infty = \gamma^{P_+} (1 - \gamma^{-k})$ - **F**'nin en büyük elemanı;

$\varepsilon_{-\infty} = -\varepsilon_\infty = -\gamma^{P_+} (1 - \gamma^{-k})$ - **F**'nin en küçük elemanı;

$\varepsilon_0 = \gamma^{P_- -1}$ - **F** kümesindeki en küçük pozitif sayı;

$1+ \varepsilon_1 = 1+\gamma^{1-k}$ - **F**'nin 1'den sonraki ilk elemanıdır.

γ , ε_∞ , ε_0 , ε_1 sayılarına **F** kümesinin karakteristikleri denir. Böylece bir Formatta P_-, P_+, k ve γ sayıları yerine ε_∞ , ε_0 , ε_1 , γ ile de yazılabilir. Açıktır ki

$$\varepsilon_\infty = \varepsilon_\infty (P_-, P_+, k, \gamma); \varepsilon_0 = \varepsilon_0 (P_-, P_+, k, \gamma); \varepsilon_1 = \varepsilon_1 (P_-, P_+, k, \gamma).$$

dir.

2.6.2. Reel Sayının Formata Yerleşme Hatası

F kümesini ve **R** reel sayılar kümesini alalım. Bu iki küme üzerinde

$$fl : [-\varepsilon_\infty , \varepsilon_\infty] \cap \mathbf{R} \rightarrow \mathbf{F}$$

şeklinde bir operatör tanımlayalım. Öyleki bu operatör, reel sayıları Format içine yerleştirsin. Eğer z bir **F** elemanı ise o zaman fl(z) = z olur. Aksi halde, eğer $z > \varepsilon_\infty$ veya $z < -\varepsilon_\infty$ ise z'e karşılık gelen Format'dan hiçbir sayı veremiyoruz, yani bu

durumda operatör tanımlanmamış oluyor. fl operatörü bilgisayarda değişik şekillerde gerçekleşmektedir, fakat her bir fl operatörünün ortak noktaları vardır.

Burada ayrıca belirtmek gerekirse, $[-\varepsilon_\infty , \varepsilon_\infty] \cap \mathbf{R}$ 'nin dışında olan sayılar bilgisayara yerleştirirken karşımıza Overflow hatası çıkacaktır. Bunun için biz bu durumda olan sayılar için fl değerini belirtmiyoruz, fakat bilgisayar kullanıcısı bu gibi durumlara düşmemek için mutlaka özel tedbirler almalıdır.

Şimdi her hangi bir z sayısını formata yerleştirdiğimizde karşımıza çıkan hatanın üst sınırını verelim. İyi biliniyor ki, fl işleminden oluşan hatanın değeri;

$$| z - fl(z) | = \alpha |z| + \beta$$

olur. Burada $|\alpha| < \varepsilon_1$, $|\beta| < \varepsilon_0$, $\alpha \beta = 0$ dir. fl işleminden sonra $|z - fl(z)|$ değerine *yuvarlama hatası (Roundoff error)* denir.

$$fl(z) = z (1+ \alpha) + \beta,$$

şeklinde yazılabilir.

2.6.3. Aritmetik İşlemlerde Hata Oluşumu , Standart Format'tan Genişletilmiş FORMAT'a Geçiş

Bilgisayar basit aritmetik işlemleri yaparken, standart Format'tan *genişletilmiş Format*'a geçiyor ve bu işlemleri genişletilmiş Format içinde yapıyor. Genişletilmiş Formatın içinde mantis için çok yer ayrılır.

Bir işlemin sonucu bilgisayarda standart formatta verildiğinde karşımıza önceki gibi reel sayının formata yerleşmesi hatası çıkar. Bu işleme giren sayıların bilgisayar sayısı dışında veya genişletilmiş formatta verilip verilmediği işlemin sonucunu etkilemez. Bu yüzden aritmetik işlemler yapılırken sonuç, tekrar dışarıdan girilen bir z reel sayısı gibi formata giriyor ve sonucun bilgisayarda gösterilişi aynı bir z reel sayısının gösterilişi gibi, bilgisayar onun yerine en yakın bilgisayar sayısını veriyor.

2.6.4. Aritmetik İşlemlerde Oluşan Hataların Üst Sınırı

Şimdi aritmetik işlemlerde hatalar için üst sınırları hesaplayalım. x, y formatın iki elemanı ise sırasıyla $x + y$, $x-y$, $x \times y$, $x \div y$, $\sqrt{x}$ işlemlerinin formattaki karşılığı $fl(x+y)$, $fl(x-y)$, $fl(x \times y)$, $fl(x \div y)$, $fl(\sqrt{x})$ olarak tanımlanır. Bu elemanter işlemler en optimal şekilde yapılırsa bu taktirde elde edilen sayılar ve gerçek sayılar arasındaki bağıntılar

$$|x + y - fl(x + y)| \leq \varepsilon_1 |x + y| + \varepsilon_0;$$
$$|x - y - fl(x - y)| \leq \varepsilon_1 |x - y| + \varepsilon_0;$$
$$|x \times y - fl(x \times y)| \leq \varepsilon_1 |x \times y| + \varepsilon_0;$$
$$|x \div y - fl(x \div y)| \leq \varepsilon_1 |x \div y| + \varepsilon_0;$$
$$|\sqrt{x} - fl(\sqrt{x})| \leq \varepsilon_1 \sqrt{x} + \varepsilon_0$$

eşitsizlikleri ile verilebilir. Bu sınırlar en optimistik sınırlardır. Bu bağıntıları formatın herhangi keyfi seçilen x ve y elemanları için veremeyiz. Her bir işlem için en az

$$|(x + y)| < \varepsilon_\infty, |(x - y)| < \varepsilon_\infty, |(x \times y)| < \varepsilon_\infty \text{ veya } |(x \div y)| < \varepsilon_\infty$$

şartları sağlanmalıdır. Böylece eğer bu şartları geçerli ise bu taktirde

$$fl(x + y) = (x + y)(1 + \alpha) + \beta,$$
$$fl(x - y) = (x - y)(1 + \alpha) + \beta,$$
$$fl(x \times y) = (x \times y)(1 + \alpha) + \beta,$$
$$fl(x \div y) = (x \div y)(1 + \alpha) + \beta,$$
$$fl(\sqrt{x}) = (\sqrt{x})(1 + \alpha) + \beta$$

eşitliklerinin doğru olduğunu kabul edebiliriz. Burada

$$|\alpha| < \varepsilon_1, |\beta| < \varepsilon_0, \alpha\beta = 0$$

dır.

Eğer z bilgisayar sayısı kayan noktalı formda verilmişse yani,

$$z = \gamma^{p(z)} \quad m(z), \quad (1/\gamma \le m(z) < 1)$$

ise, γ^p bilgisayar sayısı ile çarpıldığında, eğer Overflow durumu yoksa, ya çarpma işlemi tam yapılmıştır veya Underflow hatası var demektir. Yani,

$$z = \gamma^{p(z)} \times m(z) \times \gamma^p = \gamma^{p+p(z)} \times m(z) .$$

Bu şekilde mantis aynı kalır. O zaman ,

$$| \gamma^p \times z - fl(\gamma^p \times z) | \le \varepsilon_0, \quad fl(\gamma^p \times z) = \gamma^p \times z + \beta, \quad |\beta| < \varepsilon_0$$

olur.

2.6.5. Reel Vektör ve Matrislerin Formata Yerleşme Hatası

F kümesini ve **R** reel sayılar kümesini ele alalım. Bu iki küme üzerinde

$$fl : [-\varepsilon_\infty , \varepsilon_\infty] \cap \mathbf{R} \rightarrow \mathbf{F}$$

operatörünü daha önce tanımlamıştık. Bir NxN tipinde $A=\{a_{ij}\}$ matrisini ve bir Nx1 tipinde $g = \{g_j\}$ vektörünü formata yerleştirdiğimiz zaman karşımıza formatın elemanlardan oluşan $fl(A) =\{fl(a_{ij})\}$ matrisi ve $fl(g) = \{fl(g_{ij})\}$ vektörü çıkar. Bu matris ve vektör için

$$\| g - fl(g) \| \le \varepsilon_1 \|g\| + \sqrt{N}\, \varepsilon_0 \quad , \| A - fl(A) \| \le \varepsilon_1 \sqrt{N}\, \|A\| + N\varepsilon_0$$

eşitsizlikleri doğrudur. Eğer $\|g\|$ ve $\|A\|$ çok küçük değilse bu taktirde genellikle

$$\| g - fl(g) \| \le \varepsilon_1 \|g\| , \| A - fl(A) \| \le \varepsilon_1 \sqrt{N}\, \|A\|$$

olduğunu kabul edebiliriz.

Ayrıca, bazı ön şartlar altında matris ve vektörler ile yapılan aritmetik işlemlerde

$$\| A+ \alpha B - fl(A+ \alpha B) \| \leq \varepsilon_1 \sqrt{N} \| A+\alpha B \|;$$

$$\| f+ \alpha g - fl(f+ \alpha g) \| \leq \varepsilon_1 \| f+\alpha g \| ;$$

$$\| Ag-f - fl(Ag-f) \| \leq \varepsilon_1 \| Ag - f \|$$

eşitsizlikleri doğrudur. Burada A, B, g, f Formatın elemanlarından oluşan matris ve vektörler, α Formatın bir elemanıdır.

2.7. İki Köşegenli Denklem Sistemlerinin Bilgisayarda Hesaplaması

Burada A iki köşegen NxN tipinde reel bir matris ve f , N boyutlu bir reel vektör olmak üzere Ax = f sistemini bilgisayarda hesaplamak için bir algoritma verelim.

2.7.1. İki Köşegenli Denklem Sistemlerinin Bilgisayarda Hesaplaması

g_i, d_i, b_j , i =1,2, ... , N, j = 2, 3, ... N reel sayılar olmak üzere

$$d_1x_1+b_2x_2 = f_1$$
$$+d_2x_2+b_3x_3 = f_2$$
$$\cdots\cdots\cdots\cdots$$
$$d_{N-1}x_{N-1}+b_Nx_N = f_{N-1}$$
$$d_N x_N = f_N$$

bir reel sistemini ele alalım.

$$D = \begin{pmatrix} d_1 & b_2 & & & \\ & d_2 & b_3 & O & \\ & & \ddots & \ddots & \\ & O & & d_{N-1} & b_N \\ & & & & d_N \end{pmatrix}, g = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N-1} \\ g_N \end{pmatrix}; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix}$$

olarak seçersek bu takdirde verilen sistemi aşağıdaki Dx = g matris vektör şeklinde yazabiliriz.

Yukarıdaki sistemi bilgisayarda hesaplamak için önce, D matrisinin singüler değerlerine dayanarak bu matrisi pratik tersleyebilen mi, değil mi olduğunu kontrol etmeliyiz Eğer D well-condition matris ise bu taktirde aşağıdaki formülleri kullanarak istenilen x vektörünü hesaplayabiliriz.

$$x_N = g_N / d_N$$
$$x_{N-1} = (g_{N-1} - b_N x_N) / d_{N-1}$$
.....
$$x_1 = (g_1 - b_2 x_2) / d_1 .$$

Açıktır ki hesaplama işlemi sürecinde vazgeçilmez yuvarlama hataların sonucunda elimize bir y vektörü geçer. Bu vektörün istenilen vektör olan x ile bağlantısı aşağıdaki paragrafta inceleniyor.

2.7.2. İki Köşegenli Sistemlerin Hesaplama Sürecinde oluşan hatalarının Sınırlandırılması

$$D = \begin{pmatrix} d_1 & b_2 & & & \\ & d_2 & b_3 & O & \\ & & \ddots & \ddots & \\ & O & & d_{N-1} & b_N \\ & & & & d_N \end{pmatrix}$$

NxN tipinde iki köşegen bir matris olmak üzere. $d_1, d_2, ..., d_N$ D matrisinin diagonal ve $b_2, b_3, ..., b_N$ D matrisinin subdiagonal elemanlarıdır.

Ayrıca $g = (g_1, g_2, ..., g_N)^T$ de bir bilgisayar vektördür.

Fl(z) bir aritmetik işlemden sonra elimize geçen bilgisayar sayısını simgeliyor. $u_1, u_2, ..., u_N$ bilgisayar sayılarının D matris ve g vektör ile aşağıdaki denklemler ile bağlı olduğunu farz edelim.

$$u_N = fl\,(g_N \div d_N);$$

$$u_{k-1} = fl\,(\,fl\,(g_{k-1} - fl\,(b_k \times u_k)\,) \div d_{k-1}\,),\ k = N, N-1, \ldots, 2.$$

•

Eğer D iyi konulmuş (well-condition) bir matris ise bu taktirde hesap edilen $u = (u_1, u_2, \ldots, u_N)^T$ vektörünü $\overline{D}u = \overline{g}$ denkleminin tam çözümü olarak kabul edebiliriz. $\overline{D}$ ve $\overline{g}$ yeterince, D ve g 'ye karşılıklı yakınlardır.

Burada standart Formatta yapılan elemanter aritmetik işlemlerde, u ve z reel sayılarının sonucunda ortaya çıkan hatalar aşağıdaki şartları sağlar:

$$\begin{aligned} u \pm z &= fl(u \pm z) + \alpha\,[\,u \pm z\,] + \beta; \quad |\alpha| \le \varepsilon_1\,;\ |\beta| \le \varepsilon_0; \\ u \div z &= fl(u \div z) + \alpha_1\,[\,u \div z\,] + \beta_1; \quad |\alpha_1| \le \varepsilon_1\,;\ |\beta_1| \le \varepsilon_0; \\ u \times z &= fl(u \times z) + \alpha_2\,[\,u \times z\,] + \beta_2; \quad |\alpha_2| \le \varepsilon_1\,;\ |\beta_2| \le \varepsilon_0; \end{aligned} \qquad (0)$$

Aşağıdaki teorem doğrudur.

Uyarı: İki köşegenli D matrisi için aşağıdaki eşitsizlikler doğrudur.

$$\max\,[\,\max_{2\le j\le N} |b_j|,\ \max_{1\le i\le N} |d_i|\,] \le \|D\| \le \sqrt{\max[2d_1^2, 2\max_{2\le j\le N}(b_j^2 + d_j^2)]}\,.$$

TEOREM 2.7.2.1. *Farz edelim ki* NxN *iki köşegen* D *pratik tersleyebilen matris* ($\mu(D) < \mu^* = 1/(10\varepsilon_1)$) *ve aşağıdaki*

$$1 \ge \max\,[\,\max_{2\le j\le N} |b_j|,\ \max_{1\le i\le N} |d_i|\,] \ge 0.5\,;$$

$$0.1\sigma_1(D)\varepsilon_\infty > \|g\| > (2\,\|D\| + 3\sqrt{N})\varepsilon_0/\varepsilon_1$$

şartlar altında bilgisayarda iki köşegenli sistem başarılı bir şekilde hesaplanır ve hesaplanan u vektörü yakın bir iki köşegenli sistemin tam çözümüdür. Yani öyle bir

$\overline{D}$ matris ve $\overline{g}$ vektör vardır ki

$$\| \overline{D} - D \| \leq 6\,\varepsilon_1 \| D \| ; \quad \| \overline{g} - g \| \leq 2\varepsilon_1 \| g \|$$

olmak üzere

$$\overline{D}\,u = \overline{g}$$

olur.

İspat. Önce $\overline{D}$ matrisinin ve $\overline{g}$ vektörünün var olduğunu ispat edelim.

Elemanter işlemlerin (0) hipotezini kullanarak (j = 1, 2, ..., N; i = 2, 3, ..., N)

$$\alpha_j, \beta_j , \xi_i, \eta_i , \delta_i, \varphi_i$$

sayıları vardır ve aşağıdaki sınırlamaları

$$|\alpha_j| \leq \varepsilon_1; \quad |\beta_j| \leq \varepsilon_0 ; \quad |\xi_i| \leq \varepsilon_1 ; \quad |\eta_i| \leq \varepsilon_0 ; \quad |\delta_i| \leq \varepsilon_1 ; \quad |\varphi_i| \leq \varepsilon_0 \qquad (1)$$

sağlar ve $u_1, u_2, ..., u_N$, matris D ve vektör g ile aşağıdaki denklemler ile bağlıdırlar.

$$u_N = g_N / d_N(1 + \alpha_N) + \beta_N , k = N, N-1, ... , 2, \qquad (2)$$

$$u_{k-1} = ((g_{k-1} - (1 + \xi_k) b_k u_k + \eta_{k-1}) (1 + \alpha_{k-1}) + \varphi_{k-1}) \frac{1+\delta_{k-1}}{d_{k-1}} + \beta_{k-1}$$

Burada (k = N, N-1, ... , 2)

$$\overline{d}_{k-1} = \frac{d_{k-1}}{1+\delta_{k-1}};$$

$$\overline{b}_k = (1 + \xi_k) (1 + \alpha_{k-1}) b_k;$$

$$\bar{g}_{k-1} = \frac{\beta_{k-1}}{1+\delta_{k-1}} d_{k-1} + \varphi_{k-1} + (g_{k-1} + \eta_{k-1})(1 + \alpha_{k-1})$$

$$\bar{d}_N = d_N ; \bar{g}_N = g_N (1 + \alpha_N) + d_N \beta_N$$

olduğunu kabul edelim. Kolayca görebiliriz ki (2) aşağıdaki şekilde yazabiliriz

$$\bar{d}_N u_N = \bar{g}_N ;$$

$$\bar{d}_{k-1} u_{k-1} + \bar{b}_k u_k = \bar{g}_{k-1} , k = N, N-1, \ldots , 2.$$

Böylece $\bar{D}$ matrisinin ve $\bar{g}$ vektörünün var olduğunu ispatlamıştık.

$$\bar{D} = \begin{pmatrix} \bar{d}_1 & \bar{b}_2 & & & \\ & \bar{d}_2 & \bar{b}_3 & O & \\ & & \ddots & \ddots & \\ & O & & \bar{d}_{N-1} & \bar{b}_N \\ & & & & \bar{d}_N \end{pmatrix} ; \bar{g} = \begin{pmatrix} \bar{g}_1 \\ \bar{g}_2 \\ \vdots \\ \bar{g}_{N-1} \\ \bar{g}_N \end{pmatrix}.$$

Şimdi g ve $\bar{g}$'nin bir birine ne kadar yakın olduğunu araştıralım.

$\bar{g}$'nin tanımına göre:

$$\bar{g}_k - g_k = \frac{\beta_k}{1+\delta_k} d_k + \varphi_k + (1 + \alpha_k) \eta_k ; k = 1, 2, \ldots , N-1;$$

$$\bar{g}_N - g_N = \alpha_N g_N + d_N \beta_N$$

dir. Dolayısıyla (1) ile birlikte

$$| \bar{g}_k - g_k | \le \frac{\varepsilon_0}{1-\varepsilon_1} | d_k | + \varepsilon_1 | g_k | + (2 + \varepsilon_1) \varepsilon_0 ; k = 1, 2, \ldots , N-1;$$

$$| \bar{g}_N - g_N | \le \varepsilon_1 | g_N | + | d_N | \varepsilon_0$$

olduğunu görüyoruz. Ayrıca

$$\| \bar{g} - g \| \leq \frac{\varepsilon_0}{1-\varepsilon_1} \sqrt{\sum_{j=1}^{N} d_j^2} + 2\varepsilon_1 \| g \| + (2 + \varepsilon_1) \sqrt{N}\, \varepsilon_0$$

dir. Aşağıdaki basit eşitsizlikleri

$$\sqrt{\sum_{j=1}^{N} d_j^2} \leq \| D \|_E \leq \sqrt{N} \| D \|$$

kullanarak

$$\| \bar{g} - g \| \leq (2 \| D \| + 3 \sqrt{N}) \varepsilon_0 + \varepsilon_1 \| g \|$$

yazabiliriz. Teoremin şartlarına göre $\| g \| > (2 \| D \| + 3 \sqrt{N}) \varepsilon_0 / \varepsilon_1$ ve dolayısıyla

$$\| \bar{g} - g \| \leq 2\varepsilon_1 \| g \|$$

olduğu açıktır.

Şimdi D ve $\bar{D}$'nin bir birine ne kadar yakın olduğunu inceleyim.

$\bar{D}$ matrisinin tanımına göre (k = 2, 3, ... , N)

$$\bar{b}_k - b_k = (\xi_k + \alpha_{k-1}) b_k + \xi_k \alpha_{k-1} b_k,$$

$$\bar{d}_{k-1} - d_{k-1} = \frac{\delta_{k-1}}{1+\delta_{k-1}} d_{k-1}, \quad \bar{d}_N - d_N = 0$$

dir. Böylece (1) ile birlikte (k = 2, 3, ... , N)

$$| \bar{b}_k - b_k | \leq (2\varepsilon_1 + \varepsilon_1^2) |b_k| ;$$

$$| \bar{d}_{k-1} - d_{k-1} | = \frac{\varepsilon_1}{1-\varepsilon_1} | d_{k-1} | ;$$

$$| \bar{d}_N - d_N | = 0$$

olduğunu yazabiliriz. Bu ise aşağıdaki basit eşitsizlikler zincirini yazmaya imkan tanır.

$$\| \overline{D} - D \| \leq \sqrt{\max[2\frac{\varepsilon_1^2}{(1-\varepsilon_1)^2} d_1^2, 2 \max_{2\leq j\leq N} (\frac{\varepsilon_1^2}{(1-\varepsilon_1)^2} d_j^2, (2\varepsilon_1 + \varepsilon_1^2)^2 b_j^2)]}$$

$$\leq \sqrt{2(2\varepsilon_1 + \varepsilon_1^2)^2} \leq \sqrt{2}(2\varepsilon_1 + \varepsilon_1^2) \leq 6\,\varepsilon_1 \| D \|$$

ve sonuç olarak

$$\| \overline{D} - D \| \leq 6\,\varepsilon_1 \| D \|$$

yazabiliriz.

Sonuç 2.7.2.1. Bu teorem bize bilgisayarda elde edilen çözümün problemin tam çözümüne yakın bir çözüm olduğu yorumunu yapmaya imkan tanıyor. Literatürde bu işlem *back-error analysis* olarak biliniyor ve Wilkinson'a aittir [2].

Uyarı. Bilgisayarda elimize geçen u yaklaşık çözüm bir yandan probleminin

$$\overline{D}u = \overline{g}$$

tam çözümüdür, diğer yandan

$$\|Du-g\| \leq 16\varepsilon_1 \mu(D) \|g\|$$

doğrudur. Gerçekten,

$$Du = \overline{D}u + (D-\overline{D})u = \overline{g} + (D-\overline{D})u$$

$$Du - g = \overline{g} - g + (D-\overline{D})u$$

olduğundan aşağıdaki eşitsizliklerini yazabiliriz.

$$\| Du - g \| \leq \| \overline{g} - g \| + \| (D - \overline{D})u \| .$$

$$= \| \bar{g} - g \| + \| (D - \bar{D}) \| \| \bar{D}^{-1} \bar{g} \|$$

$$= \| \bar{g} - g \| + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \bar{g} \|$$

$$\leq \| \bar{g} - g \| + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} [\bar{g} - g + g] \|$$

$$\leq \| \bar{g} - g \| + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \| [\| \bar{g} - g \| + \| g \|]$$

$$\leq \| \bar{g} - g \| [1 + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \|] + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \| \| g \|$$

eşitsizliklerini doğru olduğunu kolayca görebiliriz. Ayrıca

$$\| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \| = \| (D - \bar{D}) \| \frac{1}{\sigma_1(D + \bar{D} - D)}$$

$$\leq \| (D - \bar{D}) \| \frac{1}{\sigma_1(D) - \| \bar{D} - D \|}$$

$$= \| (D - \bar{D}) \| / \| D \| \frac{\| D \|}{\sigma_1(D) - \| \bar{D} - D \|}$$

$$= \| (D - \bar{D}) \| / \| D \| \frac{\| D \| / \sigma_1(D)}{1 - \| \bar{D} - D \| / \| D \| \| D \| / \sigma_1(D)}$$

$$= \| (D - \bar{D}) \| / \| D \| \frac{\mu(D)}{1 - \| \bar{D} - D \| / \| D \| \mu(D)}$$

Biz biliyoruz ki;

$$\| (D - \bar{D}) \| / \| D \| \mu(D) \leq 6 \varepsilon_1 \mu(D)$$

$$16 \varepsilon_1 \mu(D) \leq 1$$

Böylece $\mu(D) \| (D - \bar{D}) \| / \| D \| < 0.5$ dir ve

$$\| \bar{D} - D \| \| [D + \bar{D} - D]^{-1} \| \leq 2 \| (D - \bar{D}) \| / \| D \| \mu(D)$$

$$\| \bar{D} - D \| \| [D + \bar{D} - D]^{-1} \| \leq 2 \| (D - \bar{D}) \| / \| D \| \mu(D) \leq 12 \varepsilon_1 \mu(D) < 1$$

olur. Dolaysıyla önceki işlemini devam olarak

$$\| Du - g \| \leq \| \bar{g} - g \| [1 + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \|] + \| (D - \bar{D}) \| \| [D + \bar{D} - D]^{-1} \| \| g \|$$

$$\leq 2 \| \bar{g} - g \| + 12\varepsilon_1\mu(D) \|g\| \leq 4\varepsilon_1 \| g \| + 12\varepsilon_1\mu(D) \|g\| \leq 16\varepsilon_1\mu(D) \|g\|$$

Dolayısıyla

$$\| Du - g \| \leq 16\varepsilon_1\mu(D) \|g\|$$

olduğu ispat edilmiştir.

2.7.3.Hesaplanan Yaklaşık Çözüm ve İstenilen Vektör İle Bağ

Önceki paragrafta görmüştük ki D iyi konulmuş matris ise Dx=g iki köşegenli sistemin bilgisayarda hesaplanması işlemi sonucunda

$$\bar{D}u = \bar{g}$$

yakın problemin tam çözümü olan, bir y vektör hesaplanır.

$$\| Du - g \| \leq 16\varepsilon_1\mu(D) \|g\|$$

olduğundan iyi bir yaklaşım elimize geçmez. Bunu klasik iterasyon işlem ile düzeltebiliriz $16\varepsilon_1\mu(D) < 1$ olsun ki. Burada biz bu işlemini yapmıyoruz. Fakat Yıldız Merdan Doktora tezinde [9] bunun için gerekli tüm incelemeleri yapmıştır.

3. DELPHI HAKKINDA BİLGİLER

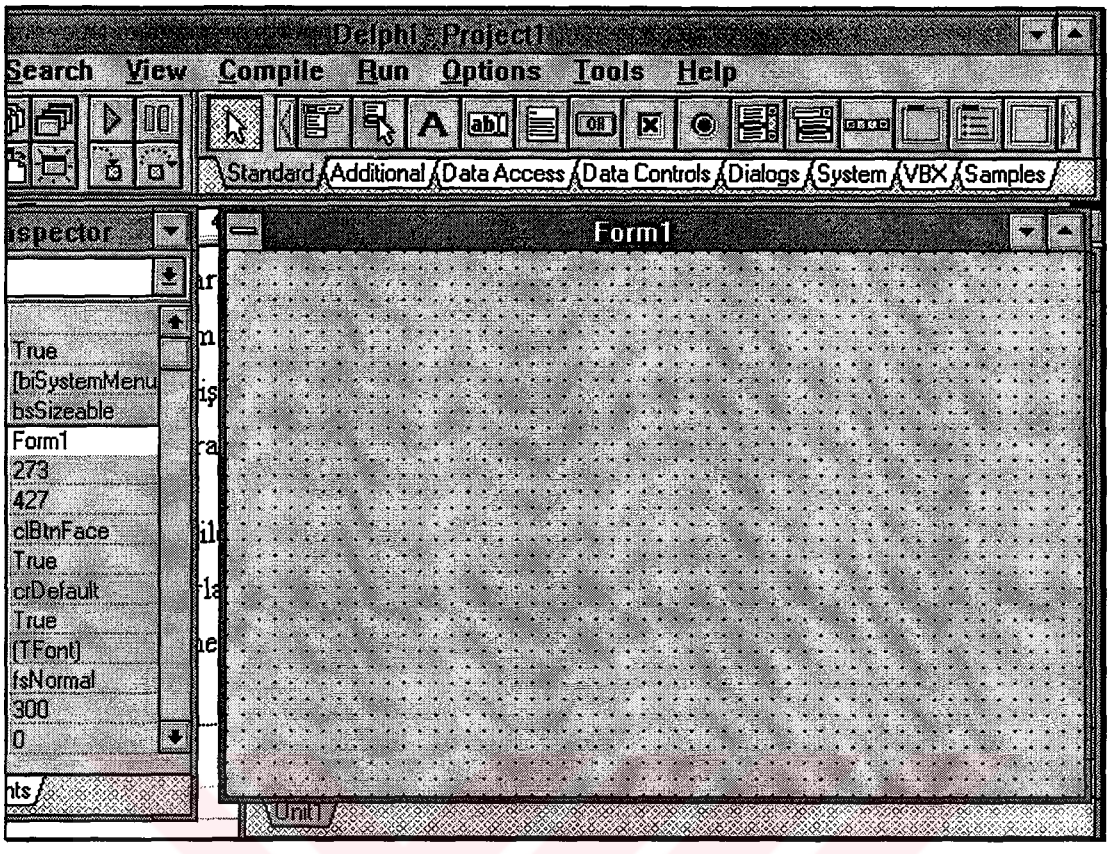
Delphi, Borland firması tarafından piyasaya sürülmüş Windows tabanlı bir program geliştirme aracıdır. Günümüzün virsual (görsel), nesne tabanlı en popüler programlarındandır. Delphi'de ana pencerenin yanında **Object Inspector** ve **form** pencereleri vardır.

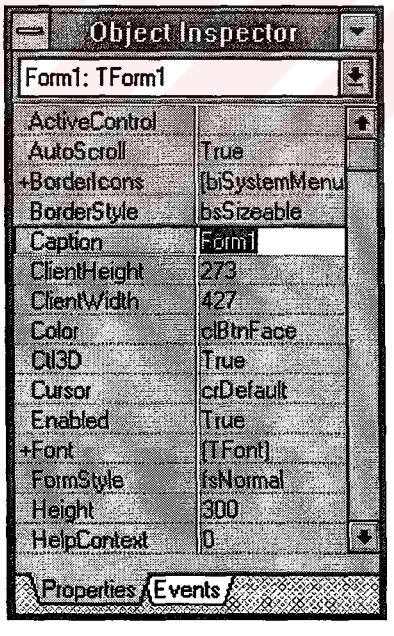
Bütün Windows uyumlu programlar ekrana bir pencere içinde gelirler ve bir pencere içinde çalışırlar. Söz konusu pencere, ekranın tümünü kaplayabileceği gibi simge durumuna da getirilmiş olabilir. Bu açıdan Windows ortamı için program yazarken, en başta, yazdığımız programın çalışması sırasında kullanıcıya nasıl bir pencere içinde yansıyacağını belirlememiz veya pencere tanımlaması yapmamız gerekir. Hatta Windows uyumlu bir program dahilinde birbirinden farklı çok sayıda pencere kullanılabilir. Örnek olarak kayıt girişi için ayrı, kayıt düzeltme için ayrı pencere kullanılabilir. Bu açıdan Windows uyumlu program yazımında ilk yapmamız gereken işlem, programın çalışmaya başladığı veya aktif olduğu zaman, kullanacağı pencereyi tanımlamaktır.

Delphi'de, diğer Windows uyumlu program geliştirme araçlarının tersine, pencerelere Window yerine form adı verilmektedir. Buna göre Delphi ile yazılan programlar, ekrana bir form içinde gelirler.

Windows uyumlu pencerelerin bazı ortak özellikleri var. Delphi ilk kez başlatıldığında hazırlanıp ekrana getirilen Form1 adındaki Form (pencere) da aynı özelliklere sahiptir. Örnek olarak Form1 adındaki formun sol üst köşesinde denetim düğmesi bulunmaktadır. Bu düğmede tıklama yapıldığı zaman, Forma ait denetim menüsü açılır.

Delphi ilk kez başlatıldığında ekrana Form1 adında bir Form hazır olarak gelir. Görüntüsü aşağıdaki gibidir.

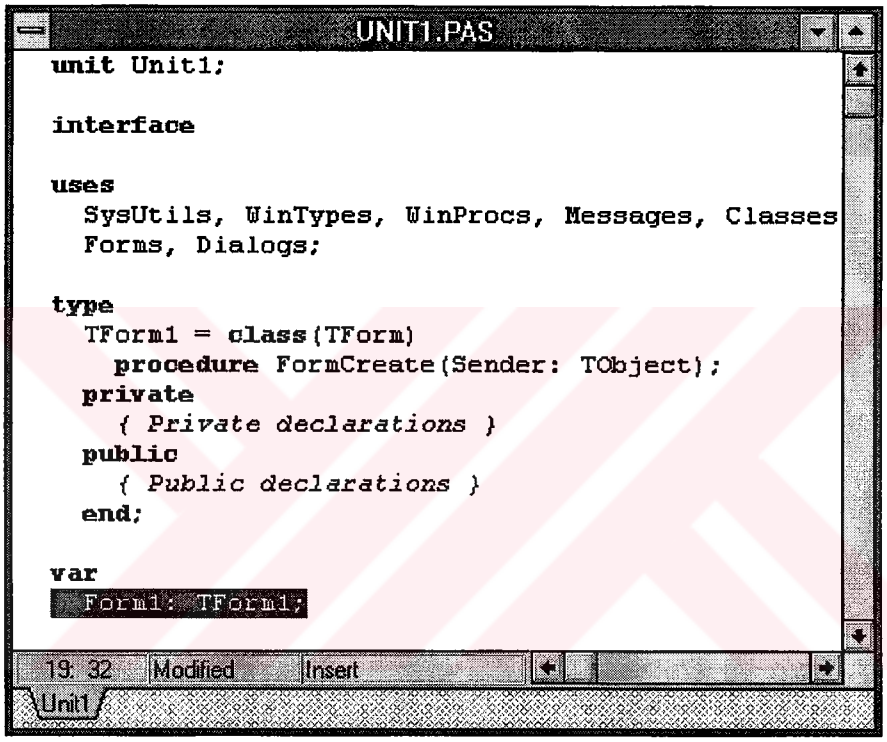




Forma ait denetim menüsünde varsayım olarak Formun kapatılması, simge durumuna gelmesi ve ekranı kaplamasını sağlayan komutlar bulunur. Delphi başladığı zaman otomatik olarak Delphi penceresine getirilen **Object Inspector** adlı

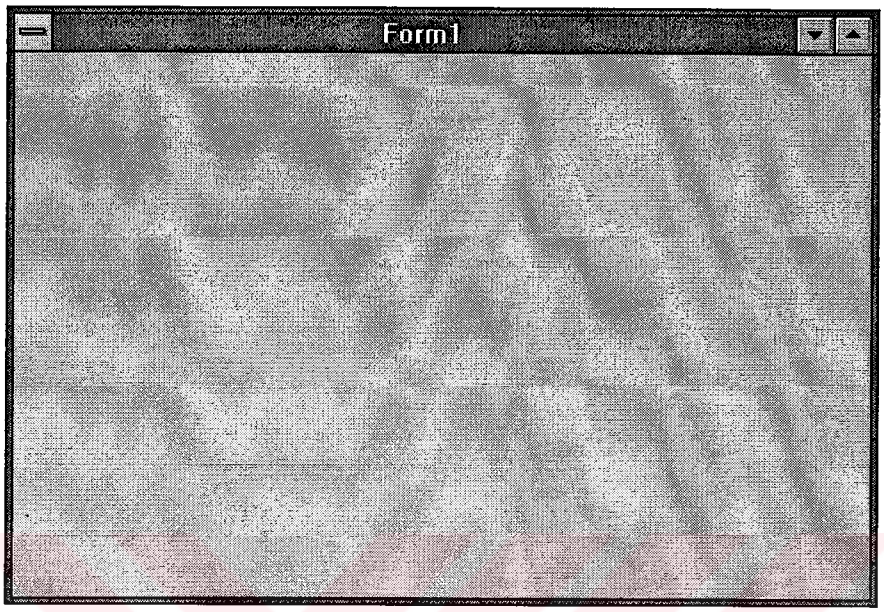
pencerede ise, formun özellikleri hakkında bilgi bulunmaktadır. Formun özelliklerini değiştirmek için bu pencereden yararlanılabilir.

Daha önceden de anlatıldığı gibi, Delphi başlatıldığında, geçici adı **Project1** olan bir proje otomatik olarak hazırlanmakta ve yine bu proje için Form1 adında bir Form otomatik olarak hazırlanmaktadır. Bunun dışında hazırlanan proje için, **UNIT1.PAS** adında Pascal program kodu içeren Bir **Unit** hazırlanmakta ve projeye dahil edilmektedir. Başlangıçta Pascal program kodu içeren bu Unit'e ait pencere, Form penceresinin altında kaldığı için masa üstünde görünmemektedir.



Bu penceredeki bütün program satırları otomatik olarak hazırlanmaktadır. Bu penceredeki program satırlarını yakından inceleyecek olursak, bu satırların bazılarının, en az özelliğe sahip olan Pascal programında olan program satırları olduğu görülür. Örnek olarak bütün Pascal programlarının başına söz konusu program dahilinde kullanılacak Unit'ler **Uses** deyimi ile programa dahil edilmektedir. Yine Pascal programları dahilinde değişkenler **Var** bildiri deyimi tanımlanmakta ve bütün Pascal programları'nın en son satırında **End** deyiminin bulunması gerekir. Delphi işimizi kolaylaştırmak için, her projede standart olarak yer alan tanımlamaları ve program satırlarını otomatik olarak hazırlamaktadır.

Aşağıda verilen ekran görüntüsünü Delphi'nin başlatılması sırasında otomatik olarak hazırlanan proje üzerinde herhangi bir işlem yapmadan, projeyi çalıştırmak için **Run** menüsünden **Run** komutunu verdikten hemen sonra alınmıştır:



3.1. Delphi'de Sayısal Değişkenler

Delphi'de matematik hesaplamalar yapmak için kullanılan çok farklı değişken türü vardır. Bu değişkenler, tamsayılar ve reel sayılar için ayrı ayrı olmak üzere önceden tanımlanmıştır.

Delphi'de tamsayılar için önceden tanımlanmış değişken türleri şunlardır:

Tür	Değer Aralığı	Formatı
Byte	0 ... 255	İşaretsiz, 1-byte
Word	0 ... 65.535	İşaretsiz, 2-byte
ShortInt	-128 ... +127	İşaretli, 1-byte
Integer	-32.768 ... +32.767	İşaretli, 2-byte
LongInt	-2147483648 ... +2147483647	İşaretli, 4-byte
Comp	-9.2 E18 ... +9.2 E18	İşaretli, 8-byte

Tamsayı türdeki operandlarla olan işlemlerde hassasiyet, değişkenin içerdiği değere göre 8-bit, 16-bit ve 32-bit hassasiyetle gerçekleştirilir.

Delphi'de bir reel tür, kayan noktalı sayıları ifade etmede kullanılan reel türlerin alt kümeler şeklinde olanıdır. Bu türler arasındaki farklılıklar; değer aralıkları, hassasiyet ve bellekte kapladıkları alan gibi özelliklerden ileri gelmektedir. Delphi'de reel sayılar için önceden tanımlanmış değişken türleri şunlardır:

Tür	Değer Aralığı	Anlamlı Hane	Büyüklük
Real	2.9 E–39 ... 1.7 E–38	11 - 12	6-byte
Single	1.5 E–45 ... 3.4 E–38	7 - 8	4-byte
Double	5.0 E–324 ... 1.7 E308	15 - 16	8-byte
Extented	3.4 E–4932 ... 1.1 E4932	19 - 20	10-byte
Comp	-9.2 E18 ... 9.2 E18	19 - 20	8-byte

Delphi, aynı zamanda; reel türdeki işlemleri yapabilmek için iki farklı makine kodu üretebilir:

- 80x87 tabanlı kayan-noktalı sayılar
- Yazılım tabanlı kayan-noktalı sayılar

Bu ikisinden hangisinin kullanılacağını $N bildirisi ile derleyiciye belirtmek gerekir. Kullanımı şöyledir:

{$N+} (Varsayılan) Kayan-noktalı işlemleri 80x87 matematik işlemcisi gerçekleştirir.

{$N-} Kayan-noktalı işlemler kütüphane fonksiyonları ile gerçekleştirilir.

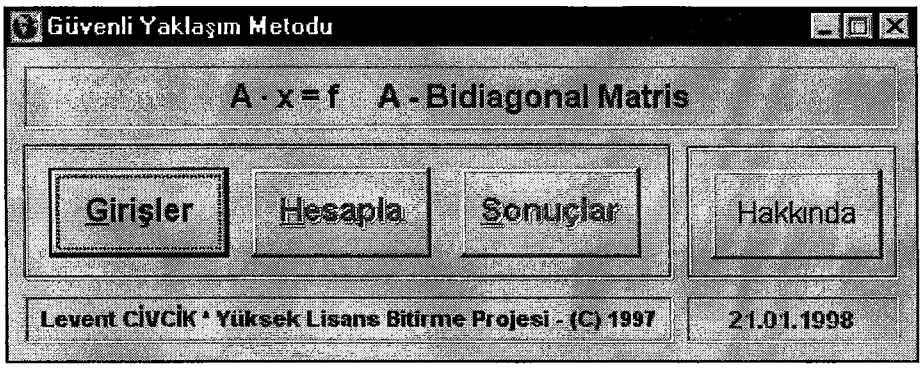
4. DİYALOG SİSTEMİNİN YAPISI

4.1. Kullanıcı ile Diyalog Sağlayan Programlar

Burada geliştirdiğimiz program, lineer denklem sistemlerinin çözümünde kullanılan Güvenli yaklaşım metodunun özel bir uygulamasıdır.

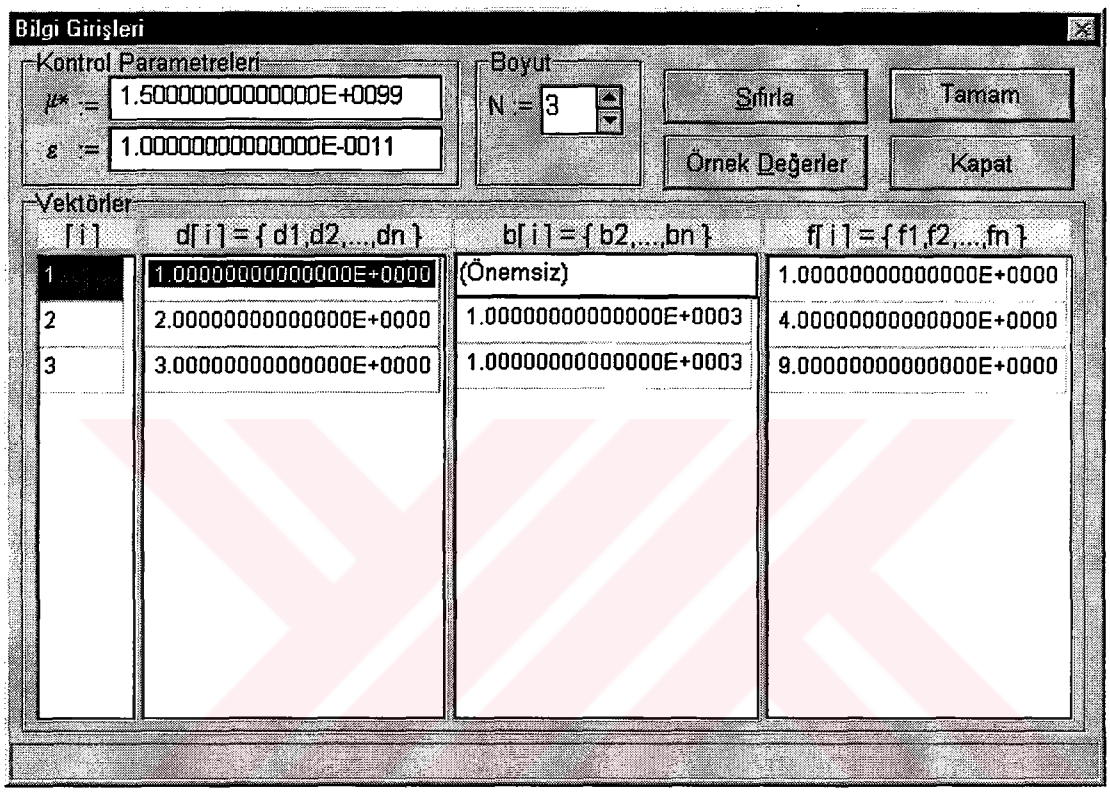
4.1.1. Ana Menü

Program başlatılınca ekrana ana form gelir. Burada işlemlerle ilgili dört düğme bulunmaktadır. Bunlar Girişler, Hesapla , Sonuçlar ve Hakkında. İlk başta bunlardan Hesapla ve Sonuçlar henüz veri girişi yapılmadığından kullanılamaz durumdadır. Girişler düğmesi tıklanınca veri girişlerinin yapıldığı yeni bir diyalog ekrana gelir. Bu durumda bir veri girişi yapılmış olduğundan Hesapla düğmesi kullanılabilir duruma gelir. Eğer Hesapla düğmesine tıklama yapılırsa klavyeden girilen ve önceden kabul edilen değerlere göre lineer denklemin çözümü ve ilgili değişkenlerin değerleri hesaplanır. Eğer hesaplama uzun sürecekse ekranda bir kum saati belirir. Hesaplama yapıldıktan sonra sonuçları tablo ve ilgili değişkenlerin değerlerini görmek için Sonuçlar düğmesi kullanılır hale gelir. Bu düğme üzerine tıklama yapılırsa Sonuçlar diyalog penceresi ekrana gelir.



4.1.2. Veri (Data) Girişleri

Burada, ilk versiyon olduğu için $\mu=1.5\cdot10^{95}$ olarak ve $\varepsilon=10^{-11}$ olarak sabit seçilmiştir. Fakat ileride bu değerlerin de kullanıcı tarafından seçilmesini amaçlıyoruz.



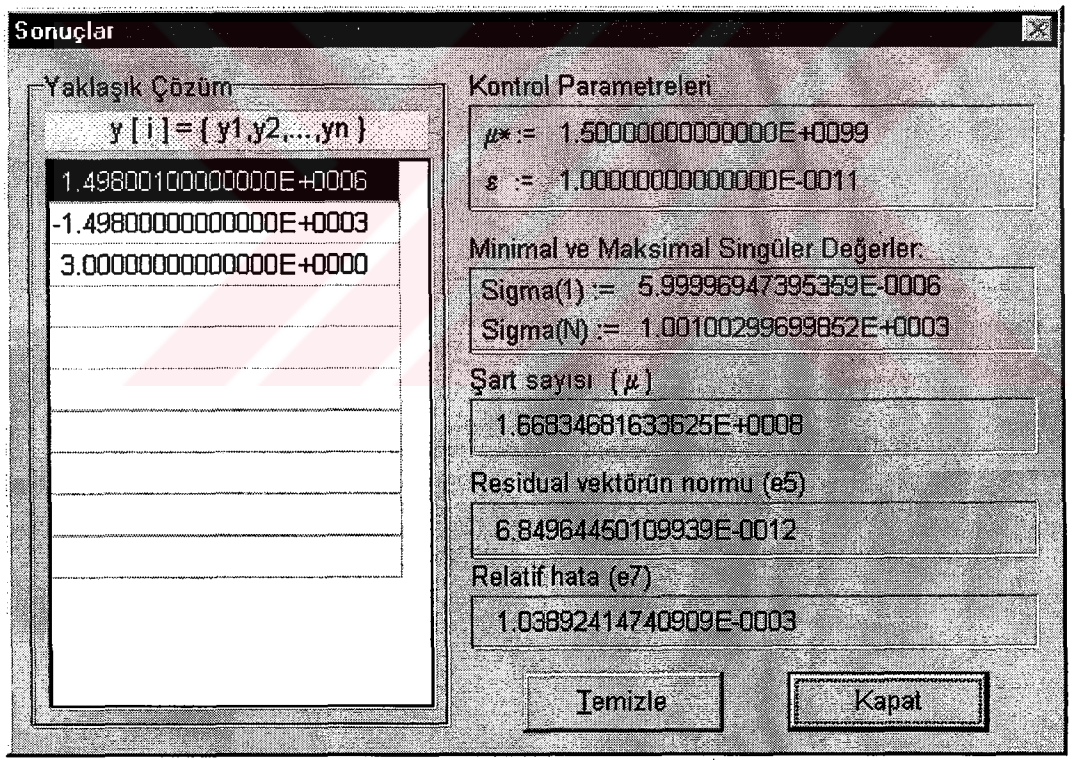
Bu pencerede problemin çözümünde temel teşkil eden boyut değiştiğinde tüm değerler ve önceden hesaplanmış sonuçlar sıfırlanmaktadır. Eğer girişlere herhangi bir değer girilmemişse bu şekilde pencereden ayrılmak mümkün değildir. Ayrıca, vektörlere kullanıcının değerleri kendisi girmesi yerine kolaylık olması için Örnek Değerler düğmesine tıklayarak rasgele değerler atanabilir yada Sıfırla düğmesine basarak tüm değerler sıfırlanabilir. Burada hesaplamalarda kullanılmadığı için b[1] değerinin hiçbir önemi yoktur. Sonraki versiyonlarda kullanılacaktır.

Kullanıcı verilerin girişini bitirince Tamam düğmesine tıklamalıdır. Tamam düğmesine tıklanınca form üzerindeki bilgiler ilgili vektörlere aktarılır ve ana menüye dönülür.

4.1.3. Hesaplamalar ve Sonuçlar

Problemin çözümü için gerekli olan değerler Veri Girişleri penceresinde girildikten sonra yaklaşık çözümleri ve ilgili değerleri hesaplamak için ana menü penceresinde Hesapla düğmesine tıklanır. Bu esnada problemin çözümü için Bölüm 2'de anlatılan problemin matematik tarafına ait işlemler gerçekleşir. Eğer işlem uzun sürerse işlemler tamamlanıncaya kadar ekranda bir kum saati belirir.

Daha sonra problemin yaklaşık çözümü olarak elde edilen sonuçları görmek için Sonuçlar düğmesine basılır. Bu diyalog penceresinde ekrana gelen bilgiler aşağıda görülmektedir. (Karakter kısıtlamaları nedeniyle semboller form üzerinde farklı olarak gösterilmişlerdir.)



Şart sayısı $\mu(A) = \|A\| \cdot \|A^{-1}\|$ dir.

Rezidü vektörünün normu (e5) $\|Ax - F\|$ değeridir.

Rölatif hata (e7) $\frac{\|x - \hat{x}\|}{\|\hat{x}\|} \leq$, burada $\hat{x}$ tam istenilen çözümdür.

5. ÖRNEKLER

$$\begin{pmatrix} \mathbf{d_1} & \mathbf{b_2} & & & \\ & \mathbf{d_2} & \mathbf{b_3} & \mathbf{O} & \\ & & \ddots & \ddots & \\ & & \mathbf{O} & \mathbf{d_{N-1}} & \mathbf{b_N} \\ & & & & \mathbf{d_N} \end{pmatrix} \begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_{N-1}} \\ \mathbf{x_N} \end{pmatrix} = \begin{pmatrix} \mathbf{f_1} \\ \mathbf{f_2} \\ \vdots \\ \mathbf{f_{N-1}} \\ \mathbf{f_N} \end{pmatrix}$$

$d_1 \cdot x_1 + b_2 \cdot x_2 = f_1$

$d_2 \cdot x_2 + b_3 \cdot x_3 = f_2$

$d_{N-1} \cdot x_{N-1} + b_N \cdot x_N = f_{N-1}$

$d_N \cdot x_N = f_N$

$$x_i = \begin{cases} \dfrac{f_i}{d_i}; i = N \\ \dfrac{1}{d_i} \cdot \{f_i - b_i \cdot x_{i+1}\}; i < N \end{cases}$$

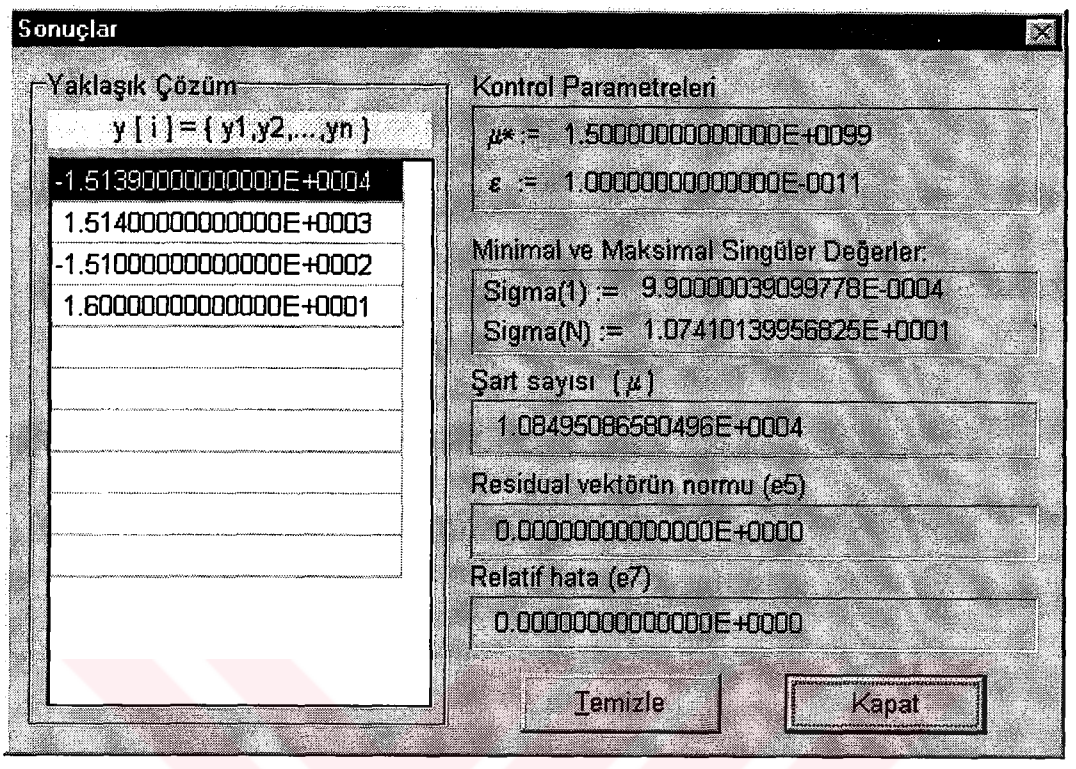
Anlaşılıyor ki; b[1] hesaplamalarda kullanılmamaktadır. f vektörünün elemanları $f[i] = i^2$ olsun.

Örnek 1:

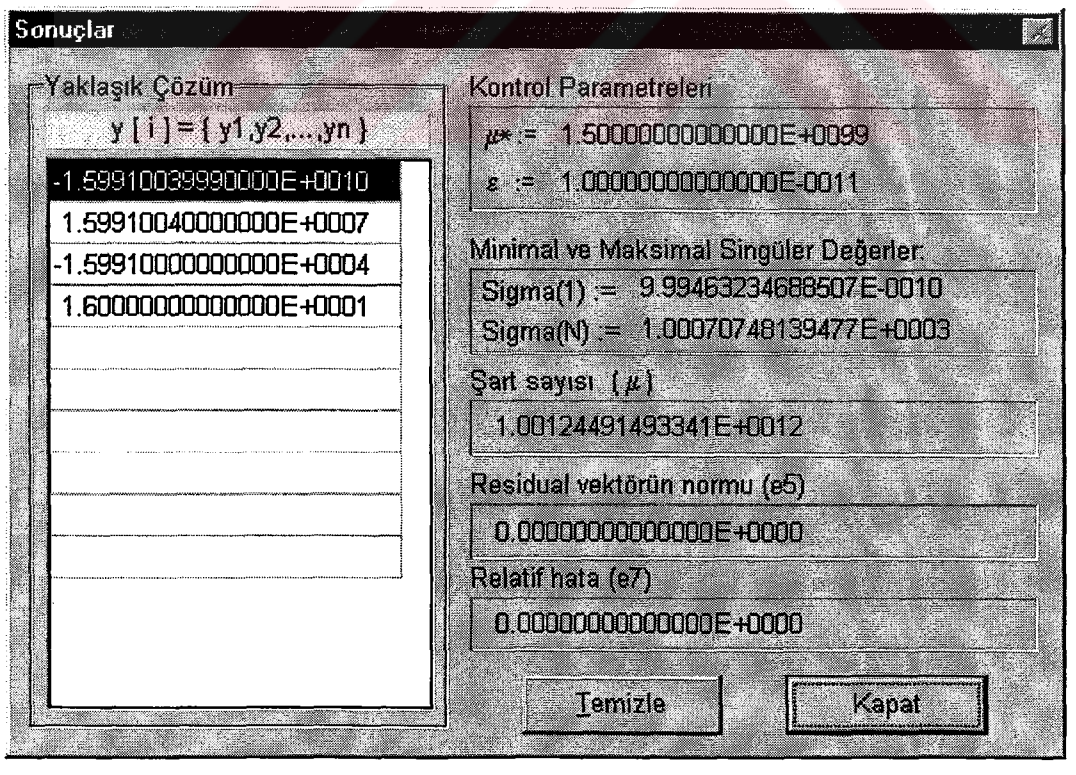
$$A = \begin{pmatrix} 1 & \alpha & 0 & 0 \\ 0 & 1 & \alpha & 0 \\ 0 & 0 & 1 & \alpha \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

matrisi ele alalım. α yerine sırasıyla 10, 1000, 10^6, 10^9 10^{40} değerlerini verelim ve bu değerlere göre hesaplamalar yaptırıp sonuçlara bakalım.

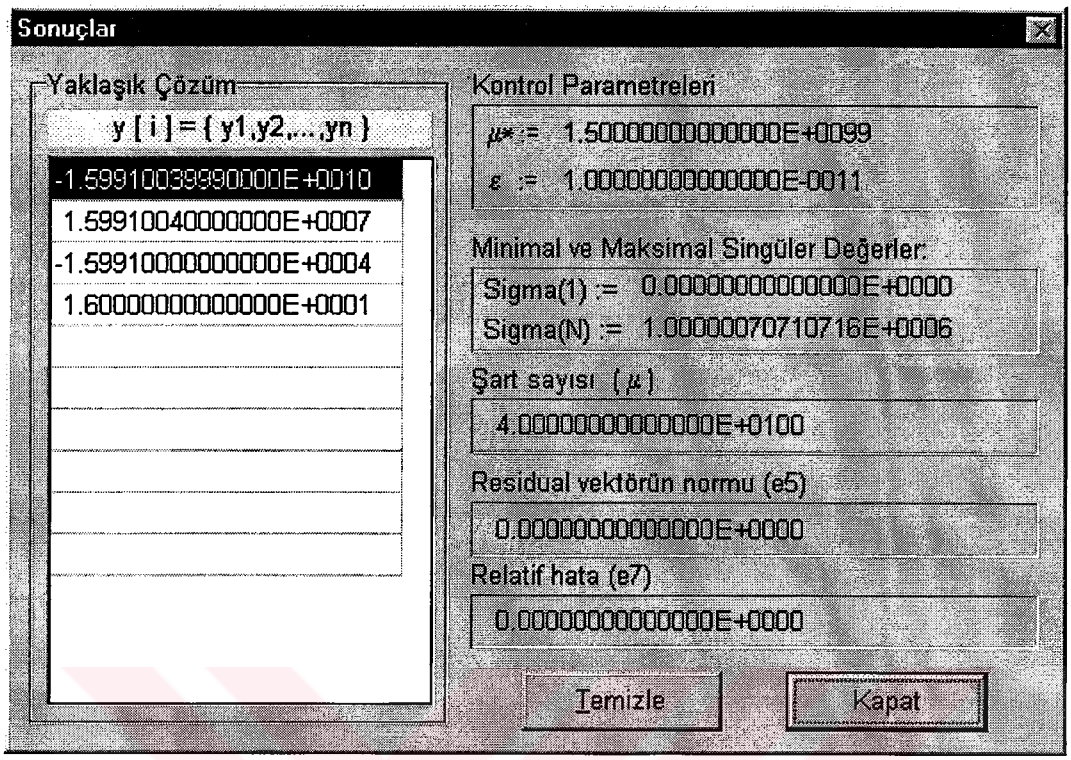
α=10 için



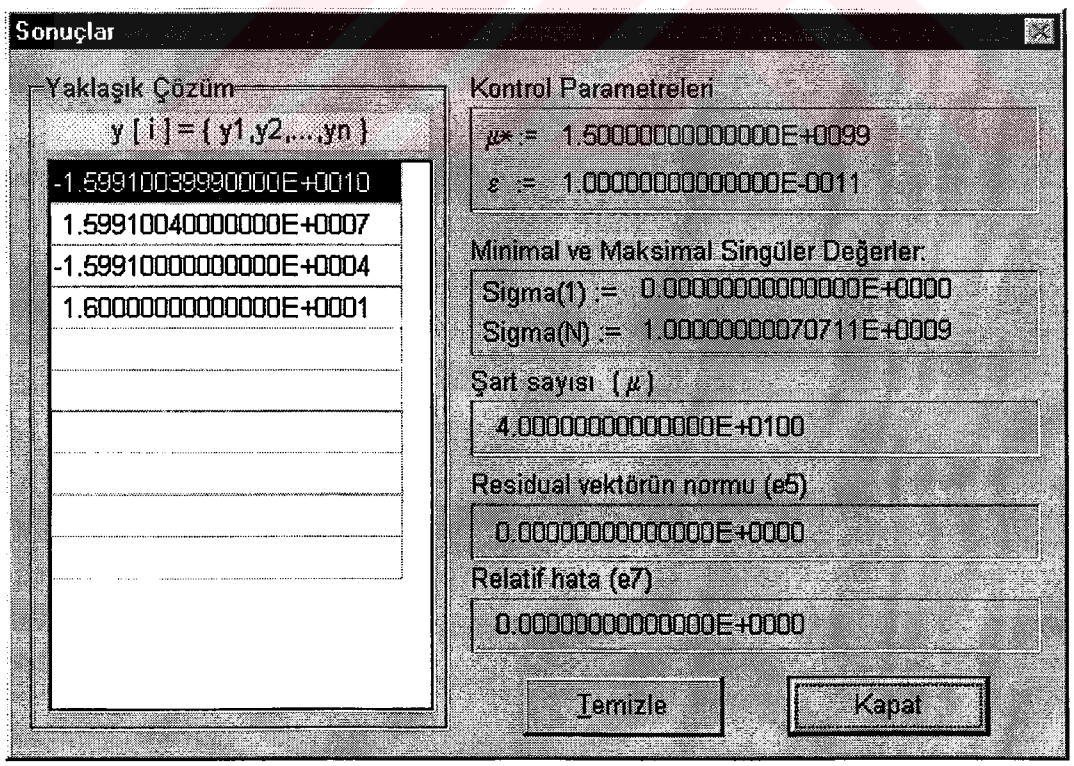
α=1000 için



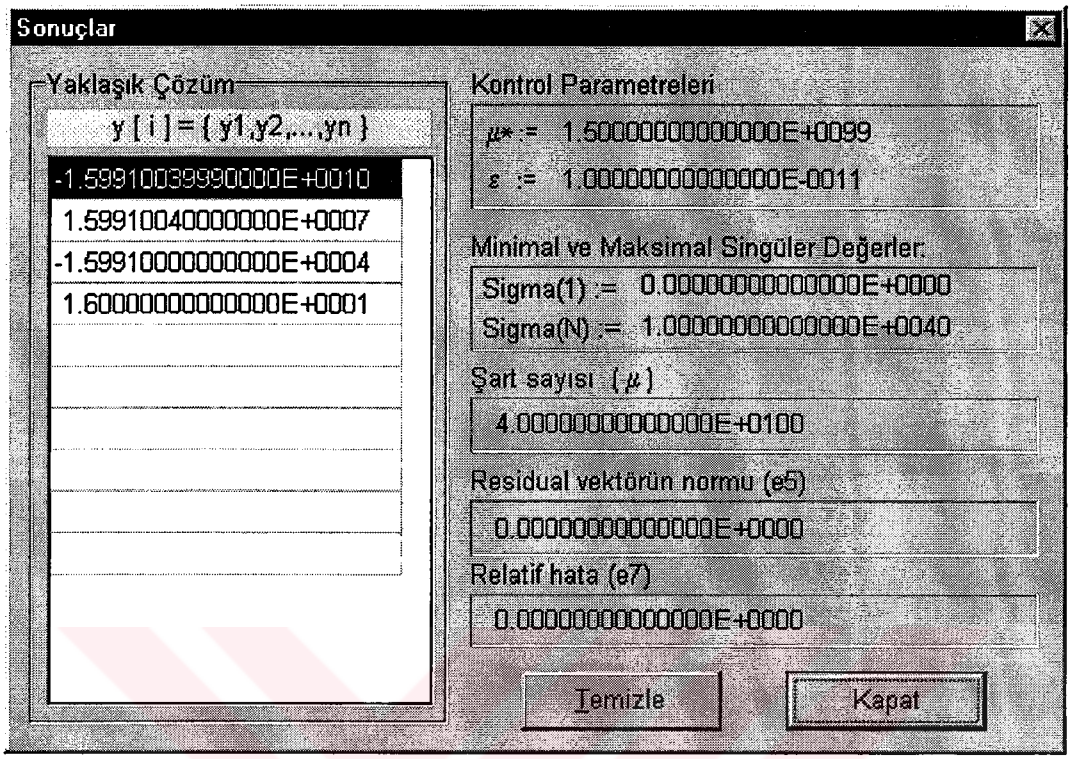
$\alpha=10^6$ için



$\alpha=10^9$ için



$\alpha=10^{40}$ için



6. DEĞERLENDİRME

Bu tez Friendly Dialogue System'in ilk versiyonunun anlatıldığı bir ön çalışmadır. Dünyada buna benzer sistemler geliştirilmeye başlanmıştır. Amacımız Selçuk Üniversitesi olarak Türkiye'de bu tür çalışmaları başlatmak ve bu çalışmaları yaygınlaştırarak, bilgisayarlı eğitim programlarına katkıda bulunmaktır.

KAYNAKLAR

1. **Godunov S.K.** (1980) Solving Linear Algebraic Systems, Novosibirsk, Nauka, (in Russian);
2. **Wilkinson J.H.** (1965) The Algebraic Problem. Clarendom Press, Oxford.
3. **Turing A.M.** Rounding-off errors in matrix processes Quart. J. Mech., 1, 1948, 287-308.
4. **J. von Neuman, Goldstine H.H.** Numerical inverting of matrices of high order. - Bull. Amer. Math. Soc., 1947, v.53, no. 11, 1021-1099.
5. **Ayşe Bulgak, Haydar Bulgak**. Lineer Cebir Sistemleri ve Garanti YaklaşımYöntemi. Selçuk University, KONYA (to appear).
6. **Godunov S.K., Antonov A.G., Kiriluk O.P., Kostin V.I.** Guaranteed accuracy for the solving of linear systems in Euclidean spaces , Novosibirsk,Nauka, 1988, -456 p. (in Russian)
7. **Ayşe Bulgak** "Bilgisayarda Sayı Gösterimi ve Hatalar", S.Ü. Fen-Bilimleri Enstitüsü, Doktora semineri, KONYA 1996
8. **Ayşe Bulgak** Lineer Cebir Sistemlerinin Şart Sayısı, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Doktora Semineri, Konya, 1997, s.23.
9. **Yıldız Merdan** "Lourier-Riccati Matris Denkleminin Garanti Yaklaşım Metoduyla Bilgisayarda Çözümü", Doktora Tezi, S.Ü. Fen-Bilimleri Enstitüsü, KONYA 1996
10. **Yanık Memik** "Borland Delphi ile Görsel Programcılık, Sistem Yayıncılık, 1996, İstanbul, ss.37-57.
11. **Klatte R., Kulish U. Wiethoff A., Lawo C. Rauch M.** C-XSC, a C++ Class Library for Extended Scientific Computing, Springer Verlag, 1993, 269 p.
12. **Bruce W. Chair, Keith O. Geddes etc.**, First Leaves: A tutorial Introduction to Maple V, Springer Verlag, 1992
13. **Gene Golub, C. Van Loan** Matrix Computations.The Joan Hopkins University Press, Second Edition, 1989, Baltimore.
14. **Oğuzer Sinan,** Friendly Dialogue System.for Solving Spectrum Portrait of a Matrix.

15. Ayşe Bulgak "Cheking of a well-conditioning of the interval matrices, Sibearian J. of Differential Equations, N.-Y., Nova Science Publ. (to appear).

[1] **Ömer Akın, Haydar Bulgak,** Lineer Fark Denklem Sistemleri ve Kararlılık Teorisi, Selçuk Üniversitesi, Konya, 1997 (basılacak), 1989, Baltimore.

[2] **Françoise Chatitin-Chatelin, Valerie Fraysse.** Lectures on Finite Precision Computations // Software-Enviroments-Tools, SIAM, Philadelphia, 1996 231 p.

[3] **Trefethen N., Bau III D.** Numerical Linear Algebra// Cornell University, 1995.

EKLER

Ek A : Programın Kaynak Kodları

Ana Projenin Kaynak Kodları

```
program M0202;

uses
  Forms,
  Master   in 'MASTER.PAS'   {MasterForm},
  Sonuclar in 'SONUCLAR.PAS' {SonucForm},
  BilgiGir in 'BILGIGIR.PAS' {BilgiGirForm},
  Bdag2    in 'BDAG2.PAS';   {Matris unit'i}

{$R *.RES}

begin
  Application.CreateForm(TMasterForm, MasterForm);
  Application.CreateForm(TSonucForm, SonucForm);
  Application.CreateForm(TBilgiGirForm, BilgiGirForm);
  Application.Run;
end.
```

Giriş Formu Kaynak Kodları

```
unit Master;

interface

uses
  Forms, SysUtils, WinTypes, WinProcs, Messages, Classes,
  Graphics, Controls, StdCtrls, ExtCtrls;

type
  TMasterForm = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    cmdGiris: TButton;
    cmdHesapla: TButton;
    cmdSonuclar: TButton;
    cmdSon: TButton;
    Panel2: TPanel;
    procedure cmdSonClick(Sender: TObject);
    procedure cmdGirisClick(Sender: TObject);
    procedure cmdHesaplaClick(Sender: TObject);
    procedure cmdSonuclarClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  MasterForm: TMasterForm;
```

```
implementation

uses BilgiGir,Sonuclar,Bdag2;

{$R *.DFM}

procedure TMasterForm.cmdSonClick(Sender: TObject);
begin
Halt;
end;

procedure TMasterForm.cmdGirisClick(Sender: TObject);
begin
BilgiGirForm.Show;
cmdSonuclar.Enabled:=False;
end;

procedure TMasterForm.cmdHesaplaClick(Sender: TObject);
begin
cmdSonuclar.Enabled:=False;
Screen.cursor:=crHourGlass;
try
   solverbr(n,mustar,eps,d,b,f,x, Lm, mu,e5, e7);
finally
Screen.cursor:=crDefault;
end;
cmdSonuclar.Enabled:=True;
end;

procedure TMasterForm.cmdSonuclarClick(Sender: TObject);
begin
 SonucForm.Show;
end;

end.
```

Bilgi Giriş Formunun Kaynak Kodları

```
unit BilgiGir;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
  Controls,Forms, Dialogs, StdCtrls, Spin, Grids,Buttons;

type
  TBilgiGirForm = class(TForm)
    cmdOK: TButton;
    GroupBox1: TGroupBox;
    lblD: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    sgd: TStringGrid;
    sgB: TStringGrid;
    sgF: TStringGrid;
    GroupBox2: TGroupBox;
    lblE: TLabel;
    lblM: TLabel;
    cmdSamples: TBitBtn;
```

```
    cmdSifirla: TBitBtn;
    GroupBox3: TGroupBox;
    seBoyut: TSpinEdit;
    lblN: TLabel;
    lblMustar: TLabel;
    lblEps: TLabel;
    procedure cmdOKClick(Sender: TObject);
    procedure seBoyutChange(Sender: TObject);
    procedure cmdSifirlaClick(Sender: TObject);
    procedure ResetAll;
    procedure SampleValues;
    procedure cmdSamplesClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  BilgiGirForm : TBilgiGirForm;

implementation

uses
   Master,Sonuclar,Bdag2;

var
  i,ecode  : integer;
  tmpS     : String[32];

{$R *.DFM}

procedure TBilgiGirForm.cmdOKClick(Sender: TObject);
label son;
begin
n:=seBoyut.Value;
for i:=0 to n-1 do
begin
if(sgD.Cells[0,i]='')then goto son else
   begin tmpS:=sgD.Cells[0,i];val(tmpS,d[i+1],ecode);end;
if(sgB.Cells[0,i]='')then goto son else
   begin tmpS:=sgB.Cells[0,i];val(tmpS,b[i+1],ecode);end;
if(sgF.Cells[0,i]='')then goto son else
   begin tmpS:=sgF.Cells[0,i];val(tmpS,f[i+1],ecode);end;
end;
BilgiGirForm.Hide;
MasterForm.cmdHesapla.Enabled:=True;
son:
end;



procedure TBilgiGirForm.seBoyutChange(Sender: TObject);
begin
n:=seBoyut.Value;
sgD.RowCount:=n;
sgB.RowCount:=n;
sgF.RowCount:=n;
end;
```

```
procedure TBilgiGirForm.cmdSifirlaClick(Sender: TObject);
begin
ResetAll;
end;

procedure TBilgiGirForm.SampleValues;
begin
n:=seBoyut.Value;
for i:=1 to n do
begin
f[i]:=i*i;str(f[i],tmpS);sgF.Cells[0,i-1]:=tmpS;
d[i]:=i;str(d[i],tmpS);sgD.Cells[0,i-1]:=tmpS;
b[i]:=1000;str(b[i],tmpS);sgB.Cells[0,i-1]:=tmpS;
end;
mustar:= 0.15e100;
eps    := 0.1e-10;
str(eps,tmpS);
lblEPS.Caption:= tmpS;
str(mustar,tmpS);
lblMustar.Caption:= tmpS;
end;


procedure TBilgiGirForm.ResetAll;
begin
n:=seBoyut.Value;
sgD.RowCount:=n;
sgB.RowCount:=n;
sgF.RowCount:=n;
for i:=1 to n do
begin
d[i]:=0;sgD.Cells[0,i-1]:='0';
b[i]:=0;sgB.Cells[0,i-1]:='0';
f[i]:=0;sgF.Cells[0,i-1]:='0';
end;
mustar:= 0.15e100;
eps    := 0.1e-10;
str(eps,tmpS);
lblEPS.Caption:= tmpS;
str(mustar,tmpS);
lblMustar.Caption:= tmpS;
end;

procedure TBilgiGirForm.cmdSamplesClick(Sender: TObject);
begin
SampleValues;
end;



procedure TBilgiGirForm.FormCreate(Sender: TObject);
begin
SampleValues;
end;

end.
```

Hesaplamaların Yapıldığı Unit'in Kaynak Kodu

```
     Unit Bdag2;
     {$N+}
     interface
     type
        vect1= array[1..101] of double;
        vect2= array[1..2]   of double;

     var
        n                 : integer;
        d,b,f,x           : vect1;
        Lm                : vect2;
        e5,e7,
        mustar,mu,eps     : double;

     procedure  solverbr(n:integer;   mustar,eps:double;d,b,f:vect1;
var x: vect1;
     var
        Lm: vect2; var mu: double; var e5: double; var e7: double);

     implementation

     type
        mat  = array[1..20]  of double;

     var
        a,a0           : mat;
        v,g,g0         : vect1;
        z,z1,maxsa     : double;
        m,norm,signal,
        nn,i,j,i5,k    : integer;


     procedure  EIVAL1(np1:integer;  d,b:vect1;  var  e:double;  var
Lm:vect2);
     var
           L,i,m,n,k,p :integer;
           e1,e2       :double;
           x,y         : vect2;
           t,r,r1,bet,r2,r3,r4,del,z,q,r5 :double;
     LABEL  LABEL35,LABEL40,LABEL55,LABEL70,LABEL85,out;
     begin
           e1:=0.15e-15;
           e2:=0.15e-320;
           n:=np1-1;
           t:=abs(d[1]);
           for k:=2 to n do begin
              r:=abs(d[k]);
              if t<r then t:=r;
              r:=abs(b[k]);
              if t<r then t:=r;
              end;
           if t>=0.5/sqrt(e2) then begin
               writeln('t>=1/(2*sqrt(e2))',t:5);
               goto out;
               end;
           b[1]:=0.;
           b[np1]:=0.;
           m:=2*n;
           r5:=2*e2;
           bet:=0;
           r:=abs(d[1]);
```

```
          for k:=2 to n do begin
                r2:=abs(d[k]);
                r1:=abs(b[k]);
                r3:=r1+r;
                if bet<r3 then bet:=r3;
                r3:=r1+r2;
                if bet<r3 then bet:=r3;
                r:=r2;
                end;
          bet:=bet*(1.+e1)+e2/8;
          del:=(1.+sqrt(3))*e1*bet+e2*(3.+bet)/8+r5*bet*bet;
          e:=3.*del;
          for k:=1 to 2 do begin
              x[k]:=0.;
              y[k]:=bet;
              end;
          i:=2;
          L:=m;
    LABEL35:
          if y[i]-x[i]>=e then goto LABEL40;
          if L<m then goto LABEL85;
          L:=np1;
          i:=1;
          y[1]:=y[2];
          goto LABEL35;
    LABEL40:
          z:=(x[i]+y[i])/2;
          q:=1.;
          p:=0;
          r1:=b[1];
          r4:=1./512;
          for k:=1 to n do begin
              r2:=d[k];
              q:=-z-r1/q*r1;
              r:=r5*(r2*r2+r4);
              if abs(q)>=r then goto LABEL55;
              if q<=0 then q:=-r else q:=r;
    LABEL55:
              if q<0.then p:=p+1;
              r3:=b[k+1];
              q:=-z-r2/q*r2;
              r:=r5*(r3*r3+r4);
              if abs(q)>=r then goto LABEL70;
              if q<=0 then q:=-r else q:=r;
    LABEL70:
              if q<0.then p:=p+1;
              r1:=r3;
       end;
          if p<L then begin
             x[i]:=z-del;
             goto LABEL35;
             end;
          y[i]:=z+del;
          goto LABEL35;
    LABEL85:
          Lm[1]:=x[1];
          Lm[2]:=y[2];
    out:
    end;

    procedure solverbr(n:integer;mustar,eps:double;d,b,f:vect1; var
x: vect1;
```

```
     var
        Lm: vect2; var mu: double; var e5: double; var e7: double);
     {
     Giris parametreleri:
     verilen A matris bi-diagonal NxN matris iki vektor ile
gosterilir:
     d(1),d(2), ..., d(N) ana diagonal;
     b(2),b(3),... , b(N) yan diagonal, !!!!
     mustar- iyi konulmus problemlerin siniri, burada 0.1e12
     olarak seçilebilir giriste;eps bu ne kadar cozume yakinlasma
hedefimiz, bu versiyonda kullanilmiyor, genelikle 0.1e-10 yetiyor;
     cikis parametreler:
     x - bir vektor istenilen cozume bir yaklasim;
     Lm(1) minimal ve lm(2) maximal A matrisinin singular degerleri;
     e5 - residual vektorunun normu ||Ax - f|| dir
     e7 - relatif hata = ||Ax - f||/||f|| dir.}

     var
     i,j,k,np1,nm1 : integer;
     s             : extended;
     v,g,g0        : vect1;
     fmax,normf,
      z,z1,e1,e2   : double;
     LABEL
           out;
     begin
           e1:=0.15e-15;
           e2:=0.15e-320;

        eival1(n+1,d,b,z,Lm);
     if Lm[1]< 0.25e-100 then begin
                              mu:=4e100;
                              goto out;
                              end;
     mu:=Lm[2]/Lm[1];
        fmax:=0;
        for i:=1 to n do
            if fmax<abs(f[i]) then fmax:=abs(f[i]);
            if fmax=0 then z:=0 else z:=1/fmax;
                        s:=0;
                    for i:=1 to n do begin
                        z1:=f[i]*z;
                        s:=z1*z1+s;
                        f[i]:=z1;
                        end;
                        normf:=sqrt(abs(s));
              x[n]:=f[n]/d[n];
              nm1:=n-1;
              for i:=1 to nm1 do begin
                  s:=f[n-i]-b[n-i+1]*x[n-i+1];
                  x[n-i]:=s/d[n-i];
                  end;

                  z1:=d[n]*x[n]-f[n];
                  e5:=z1*z1;
                  nm1:=n-1;
                  for i:=1 to nm1 do begin
                      s:=f[n-i]-b[n-i+1]*x[n-i+1]-x[n-i]*d[n-i];
                      e5:=e5+s*s;
                  end;
                  e5:=sqrt(abs(e5));
                  e7:=e5/normf*mu;
```

```
            for i:=1 to n do
                x[i]:=x[i]*fmax;
out:
end;

end.
```

Sonuçların Gösterildiği Formun Kaynak Kodları

```
unit Sonuclar;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
  Controls, Forms, Dialogs, ExtCtrls, Grids, StdCtrls;

type
  TSonucForm = class(TForm)
    cmdOK: TButton;
    GroupBox1: TGroupBox;
    lblX: TLabel;
    sgX: TStringGrid;
    lblSing: TLabel;
    Bevel1: TBevel;
    lblLm1: TLabel;
    lblLm2: TLabel;
    Label1: TLabel;
    Bevel2: TBevel;
    lblMu: TLabel;
    Label2: TLabel;
    Bevel3: TBevel;
    lble5: TLabel;
    Label3: TLabel;
    Bevel4: TBevel;
    lble7: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    cmdTemizle: TButton;
    procedure cmdOKClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure cmdTemizleClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  SonucForm: TSonucForm;

implementation

uses Master,BilgiGir,Bdag2;

var
  i,ecode   : integer;
  tmpS      : String[32];

{$R *.DFM}

procedure TSonucForm.cmdOKClick(Sender: TObject);
```

```
begin
SonucForm.Hide;
end;

procedure TSonucForm.FormActivate(Sender: TObject);
begin
for i:=0 to 9 do begin
   sgX.Cells[0,i]:='';
end;
lblLm1.Caption:='';
lblLm2.Caption:='';
lblMu.Caption:='';
lble5.Caption:='';
lble7.Caption:='';

for i:=0 to n-1 do begin
     str(x[i+1],tmpS);sgX.Cells[0,i]:=tmpS;
end;
str(Lm[1],tmpS);lblLm1.Caption:=tmpS;
str(Lm[2],tmpS);lblLm2.Caption:=tmpS;
str(mu,tmpS);lblMu.Caption:=tmpS;
str(e5,tmpS);lble5.Caption:=tmpS;
str(e7,tmpS);lble7.Caption:=tmpS;
end;

procedure TSonucForm.cmdTemizleClick(Sender: TObject);
begin
tmpS:='';
for i:=0 to n-1 do begin
     x[i+1]:=0;sgX.Cells[0,i]:=tmpS;
end;
Lm[1]:=0;lblLm1.Caption:=tmpS;Lm[2]:=0;lblLm2.Caption:=tmpS;
Mu:=0;lblMu.Caption:=tmpS;e5:=0;lble5.Caption:=tmpS;
e7:=0;lble7.Caption:=tmpS;
MasterForm.cmdSonuclar.Enabled:=False;
end;
end.
```