1. GİRİŞ

Hemen hemen her bilim dalında olduğu gibi Sayısal Çözümlemenin (Nümerik Analizin) sınırları da kesin olarak belirlenememektedir. Bilgisayarların gelişmesi ve yaygınlaşmasıyla beraber Sayısal Çözümleme çok geniş uygulama alanı bulmuştur. Sayısal Çözümleme ; Bilgisayar Bilimi (Bilgisayar Mühendisliği) ve Uygulamalı Matematikten kaynak almakta ve bu bilim dallarından aldıklarını işleyerek çok değişik disiplinlere başta mühendislikler olmak üzere fen bilimleri,istatistik, işletme, ekonomi,sosyal bilimler ve davranış bilimlerine hizmet vermektedir. Sayısal Çözümleme tanım olarak bilgisayar bilimiyle uygulamalı matematiğin ortak bir dalıdır. Sayısal Çözümlemenin amacı ; matematik modellerle ifade edilmiş çok çeşitli alanlara ait problemlerin çözümünde belli sayıda ve sırası belirlenmiş işlemleri uygun bir hesaplayıcı (bilgisayar) yardımıyla yaparak belirli bir duyarlığa sahip sonuçlar elde etmek için kullanılabilecek yöntemlerin bulunması, geliştirilmesi, var olanların irdelenmesi ve en etkin olanların saptanması olduğu söylenebilir. Bir problemin çözümü için kullanılabilecek çeşitli yöntemler (algoritmalar) olabilir. Sayısal Çözümlemenin bir amacı da en az işlem süresi, en az bellek gerektiren ve en duyarlı sonuçları veren algoritmaların araştırılmasıdır.

Genel de karşılaşılan problemlerin çözümünün kesin ve yaklaşık olarak iki grupta olduğu söylenebilir. Grafik çözümlerde hiç hesap yapmadan çizim suretiyle sonuc elde edildiğini biliyoruz. Ancak grafik çözümün yaklaşık bir çözüm olduğunu ve duyarlığının düşük olacağını da unutmamak gerekir. Kesin çözümler analitik çözümlerdir. Problemin kesin çözümü verilere bağlı olarak direkt ifade edilebilir. Bazı problemlerin kesin çözümü olmadığı da bir gerçektir. Bazı integral hesaplamalarının analitik olarak gerçekleştirilememesi nedeniyle örneğin normal dağılım eğrisi altında kalan alan (olasılık) hesaplamaları kesin olarak yapılamamakta ancak yaklaşık olarak ifade edilebilmektedir. Özellikle kesin çözümü yapılamayan problemler sayısal çözümleme teknikleriyle yeterli doğruluk verecek şekilde çözülebilmektedir.

2. BİLGİSAYARLARLA PROBLEM ÇÖZÜMÜNDE EN UYGUN (OPTİMUM) PROGRAMLAMA

Bilgisayarların hesaplama alanında kullanılması hesaplamalara çok büyük kolaylıklar getirmiştir. Yakın geçmişten günümüze kadar bilgisayarların hız ve bellek kapasitelerinde müthiş artış olduğu gözlenmiş ve bu artışın önümüzdeki yıllarda da devam edeceği tahmin edilmekltedir. Bilgisayarlardaki bu performans artışına paralel olarak insanoğlunun ilgilendiği problemlerin boyutuda benzer şekilde büyümektedir. Örneğin gen araştırmaları için bugünün süper diye adlandırılan bilgisayarlarının bile aylarca durmadan çalışması gerekeceği söylenmektedir. Dolayısıyla her zaman bazı özel hesaplamalar için bilgisayarların hız ve kapasitelerinin yetersiz kalabileceği söylenebilir ve bu nedenlerle problem çözümlerinde optimum programlama tekniklerinin uygulanması gerekir.

Optimum programlama, çözümde en uygun yöntem seçimi ve çalışma süresini azaltan bir takım programlama teknikleriyle yapılır.

Optimum programlama için seçilen ideal bir yöntemin aşağıdaki dört koşulu sağlaması gerekir.

1- Yöntemin kolay ve kısa programlanabilir olması
2- Yöntemin en az çalışma süresini gerektirmesi
3- Yöntemin en az belleği gerektirmesi
4- Yöntemin en duyarlı (hassas) sonuçları vermesi

Bu dört koşulun tümünün sağlanması ideal bir durum olup, genellikle yöntemlerin hiç biri bu dört koşulun tümünü sağlamaz. Böylesi durumlarda hangi koşul ya da koşulların sağlanması gerektiğine kullanıcı karar verir.

2.1 YÖNTEMİN KOLAY VE KISA PROGRAMLANABİLİR OLMASI

Bu koşul ön plana alındığında genellikle diğer koşullar sağlanmaz. Örneğin doğrusal denklem sistemlerinin çözümünde kullanılan Gauss, Gauss-Jordan, Cramer ve Cholesky yöntemlerinden Cramer yönteminin algoritması çok basit olmakla beraber diğerlerine oranla çok daha fazla çalışma süresi gerektirmekte ayrıca duyarlı (hassas) sonuçlar da vermemektedir.

2.2. YÖNTEMİN EN AZ ÇALIŞMA SÜRESİNİ GEREKTİRMESİ

Programın gerektireceği çalışma süresi bir yerde seçilen yönteme bağlıdır. Örneğin doğrusal denklem sistemlerinin çözümünde Gauss yöntemi diğerlerine göre en hızlıdır. Problemlerin dolaylı (iteratif) çözümlerinde genellikle hesaplama süresi fazladır. Çok ender de olsa bazen dolaylı çözümlerin direkt çözümlerden daha kısa zaman aldıkları görülebilmektedir. Yukarıda programın gerektireceği çalışma süresinin seçilen yönteme bağlı olduğunu söylemekle beraber, program içinde yapılabilecek bir takım düzenlemeler ile çalışma süresi en aza indirilebilir. Şimdi bunlardan bazılarını görelim.

Sadeleştirme :

$x = ab + ac$

şeklindeki bir aritmetik işlem aşağıdaki gibi iki farklı şekilde yazılabilir.

10 x = a*b + a*c $\Rightarrow$ 10 x = a*(b+c)

Şüphesiz ikinci yazım şekli (yukarıda sağdaki) doğru olanıdır. Böylece hesaplayıcıya iki çarpma bir toplama işlemi yerine bir çarpma bir toplama işlemi yaptırılmış olur.

Uygun Duyarlıklı Sayılarla Çalışma :

Bilgisayarlarda çift duyarlıklı sayı yerine tek duyarlıklı sayı,tek duyarlıklı sayı yerine tam sayı kullanma bellek gereksinimini ve çalışma süresini azaltır.

Örneğin : a = 3b şeklindeki bir çarpma işlemi

1-) a = 3! * b (çift duyarlıklı)

2-) a = 3. * b (tek duyarlıklı)

3-) a = 3 * b (tamsayı)

yukarıdaki üç şekilde de program içinde kullanılabilir. Kuşkusuz bunlardan üçüncüsü programlama tekniği yönünden en uygun olanıdır.

Uygun İşlem Seçimi :

Bazı durumlarda aritmetik işlemler birbirleri yerine kullanılabilmektedirler. İşlem süresi yönünden sıralama

toplama < çıkarma < çarpma < bölme

şeklindedir.

Buradan çıkarma yerine toplamayı , bölme yerine de çarpmayı kullanarak işlem (hesap) süresini azaltmış oluruz.

*** Bölme işlemi yerine Çarpma işlemini kullanma**

10 x = a / 2 şeklindeki bir deyimi

10 x = a *.5 şeklinde yazmak daha uygun olur.

*** Bir sayıyı -1 ile çarpmak yerine eksi olarak atamak**

10 x = -1 * a yerine

10 x = -a

*** Üs hesabını çarpım şeklinde yapmak**

10 x = a^2 yerine 10 x = a * a

20 x = a^3 yerine 20 x = a * a * a

*** Olabildiğince üs hesabından kaçınmak**

$x = a^2 - b^2$ şeklindeki bir işlemi
30 x = a^2 - b^2

şeklinde veriliş biçimine uygun olarak yazılabilir. Ancak söz konusu deyimi

30 x = (a+b)*(a-b) ya da

30 x = a*a - b*b

şeklinde yazmak daha uygun olur.

*** 0.5. üs yerine hazır karekök fonksiyonunu (SQR) kullanmak**

```
40 x = a^0.5      yerine    40 x = SQR(a)
```

yoluyla hesaplama süresi azaltılabilir.

Bilgisayar üs alırken, verilen açıların trigonometrik değerlerini bulurken ya da logaritma alırken seri açınımı yapar. Bu nedenle bu tür hesaplamalardan olabildiğince kaçınmak, işlem süresi azaltma yönünden uygun olur.

Örneğin verilen bir açının sinüs, kosinüs ve tanjant değerlerinin hesaplandığı aşağıda soldaki program parçası

```
10  sin = SIN(X)          10 sin = SIN(X)
20  cos = COS(X)          20 cos = SQR(1-sin*sin)
30  tan = TAN(X)          30 tan = sin/cos
```

yerine sağdaki biçimiyle kullanılması uygun olur.

*** Simetrik bir matrisin tümünün hesaplatılması yerine yalnızca köşegen ve köşegen üstündeki terimlerin hesaplatılması**

Örnek olarak elemanlarının değeri satır ve sütun indislerinin çarpımı olan matrisin oluşturulmasını inceleyelim. Aşağıda sağdaki programda olduğu gibi simetrik elemanlar için atama işlemi yapılırsa hesaplama süresinden tasarruf edilmiş olur.

```
10 for i = 1 to n          10 for i = 1 to n
20 for j = 1 to n          20 for j = i to n
30 a(i,j)= i*j             30 a(i,j)= i*j
40 next j                  35 a(j,i)= a(i,j)
50 next i                  40 next j
                           50 next i
```

*** Mantıksal ya da Aritmetik IF komutunun kullanımından olabildiğince kaçınmak**

Mantıksal ya da aritmetik IF komutunun kullanılarak yapılan karşılaştırmalar önemli bir işlem zamanı alırlar bu nedenle IF deyimi kullanırken dikkatli olunmalı gereksiz yere kullanılmamalı eğer, IF deyimi bir döngü içinde geçiyorsa çok daha dikkatli davranılmalıdır. Örneğin yukarıdaki sağdaki program parçasında 35 nolu deyimin doğrusu

```
35 if (i< >j) a(j,i)= a(i,j)
```

şeklinde olmalıdır. Ancak bu haliyle sözkonusu IF deyimi n x n kez yürütülecektir. Oysa 35 nolu deyimin eski halinde if komutu geçmemekte, fakat matrisin köşegen

elemanları için toplam n kez gereksiz atama işlemi yapılmaktadır. Programın bu haliyle kullanılması sadelik ve hız açısından avantajlıdır.

*** Etkin indis kullanmak**

Dizinli değişkenlerde (vektör, matris, üçboyutlu matris, ...) boyut sayısı arttıkça bellekteki bilgiye ulaşma zamanı artar. Bu nedenle etkin indis kullanılarak çok boyutlu diziler tek boyutlu dizilere dönüştürülürek çalışma süresi azaltılır. Bilgisayar belleğinde bir skalara ulaşmak dizinli değişkenlere ulaşmaktan daha çabuk olmaktadır. Özellikle matris çarpmalarında buna dikkat etmek gerekir.

Örneğin : $C_{(n,m)} = A_{(n,r)} * B_{(r,m)}$ matris çarpımı klasik olarak aşağıda solda yazılmış program parçasıyla gerçekleştirilmektedir.

```
10 for i = 1 to n                    10 for i = 1 to n
20 for j = 1 to m                    20 for j = 1 to m
30 c(i,j)= 0                         30 t = 0
40 for k = 1 to r                    40 for k = 1 to r
50 c(i,j)=c(i,j)+a(i,k)*b(k,j)       50 t=t+a(i,k)* b(k,j)
60 next k                            60 next k
70 next j                            70 c(i,j)=t
80 next i                            80 next j
                                     90 next i
```

Yukarıda solda 50 nolu deyimde n x m x r kez bellekte indis hesabı yapılarak, C matrisinin ilgili değerine ulaşılmaktadır. Oysa sağdaki programda herhangibir indis hesabı yapmadan t skalarına ulaşılarak çalışma süresi azaltılmaktadır.

2.3. YÖNTEMİN EN AZ BELLEĞİ GEREKTİRMESİ

Programın gerektireceği bellek çoğu kez seçilen yöntemle doğrudan bağlantılıdır. Örnek olarak ters matris hesaplamaya yarayan bir dolu yöntem vardır. Örneğin bazı ters matris algoritmaları bir ya da iki tane yardımcı matrise gerek duyarken bazı algoritmalar ise ters matris hesabını giriş matrisi üzerinde yapabilmektedir. Özellikle tersi hesaplanacak matris simetrik ise kullanılacak ters matris programının simetrik matrisler için yazılan program olması uygun olur. Böylelikle tersi hesaplanacak giriş matrisinin üst üçgen kısmı bir tek boyutlu diziye (vektöre) yüklenir. Ters matris hesabıda aynı dizi üzerinde yapılarak minimum bellekte ters matris hesabı gerçekleştirilmiş olur. Aynı durum band yapılı denklem sistemlerinin çözümünde de söz konusudur. Bu tür denklem sistem çözümlerinde band yapıya uygun algoritma kullanılmaması gereksiz yere bellek kullanımına neden olur. Ayrıca programın gerektireceği bellek kullanılan değişken türlerine de bağlıdır. Sayısal bilgiler türlerine göre bellekte farklı uzunlukta alanlara yerleştirilirler.

1	Tamsayı (Integer) Bilgi	2 byte
1	Gerçel (Reel) Bilgi	4 byte
1	Çift Duyarlıklı (double precision) Bilgi	8 byte

Bu nedenle programda uygun duyarlıkta sayılarla çalışmak gerekir. Böylece hem çalışma süresinden hem de bellekten tasarruf sağlanmış olur.

Bu konuya çarpıcı bir örnek olarak , Bilgisayar programlarında oldukça sık kullanılan döngüler (FOR-NEXT) de sayaç değişkenlerinin tamsayı tanımlanmasını verebiliriz.

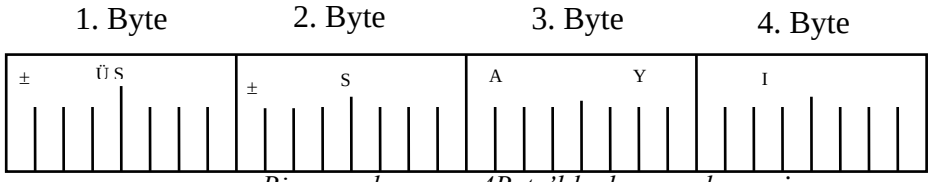
```
10 for i = 1 to n        10 for i% = 1 to n
20 x(i) = 0              20 x(i%) = 0
30 next i                30 next i%
```

Yukarıda soldaki program parçasında n elemanlı bir x dizisinin klasik olarak sıfırlanması gösterilmektedir. Bu işlem sağdaki şekliyle yapıldığında çalışma zamanı azalacaktır.

2.3.1. SAYISAL BİLGİLERİN BELLEKTE SAKLANIŞ BİÇİMLERİ

Bilgisayar belleğinin en küçük birimi "Bit" dir. 1 Bit' lik alana yalnız bir karakter 0 ya da 1 yazılabilmektedir. Tüm bilgiler (sayısal, alfa sayısal, görüntü,ses) bigisayarda 0,1 kodlarına dönüştürülerek saklanır ve işlem görürler. Sayısal bilgiler de işte bu bit gruplarına yerleştirilmektedir. 8 Bit'lik alan 1 Byte olarak adlandırılmaktadır. Şimdi bir gerçel sayının nasıl saklanacağını görelim. Bilindiği gibi gerçel sayılar 4 Byte' lik alanlara yerleştirilmektedirler.

1 Byte = 8 Bit
4 Byte = 32 Bit



Bir gerçel sayının 4Byte'lık alana yerleşmesi

Yukarıdaki şekilden görüldüğü gibi 1. byte sayının üssüne ayrılmakta, 1. byte'in 1. bit'i de üssün işaretine tahsis edilmekte, geri kalan 2.,3. ve 4.byte'lar sayının kendisine, bu byte' lar dan 2.Byte'in 1. bit'i sayının işaretine ayrılmaktadır. Buradan sayının kendisine 23 bit'lik bir alan ayrıldığı görülmektedir. Bu alana yazılabilecek en büyük sayısal bilgi, sayıların ikili sayı sisteminde yerleştiği göz önünde bulundurulursa 2^{23}-1= 8388607 olacaktır. Bu, bundan daha büyük sayıların saklanamayacağı anlamına gelmez. Bu sayıdan daha büyük sayılar üslü olarak saklanırlar. Buradan çıkan anlam gerçel sayılardaki güvenilir basamak sayısının 7 olduğudur.

Tamsayı bilgiler 2 byte'lık alanlarda 15 bit'e yerleştirilir. Bu alana yerleştirilecek en büyük tamsayı 2^{15}-1 = 32767 dir.

Çift duyarlıklı sayılar 8 byte'lık alanlarda 55 bit üzerine yerleştirilir. Bu alana yerleştirilecek en büyük bilgi 2^{55}-1 = 3.6028797*10^{16} dır. Çift duyarlıklı sayılarda güvenilir basamak sayısı 16 dır.

	Bellekteki uzunluğu	Alabileceği değer (min,max)	Güvenilir Basamak Sayısı
Tamsayı	2 Byte	-32768, 32767	5
Gerçel	4 Byte	-10^{78}, 10^{77}	7
Çift Duyarlıklı	8 Byte	-10^{78}, 10^{77}	16

2.4. YÖNTEMİN EN DUYARLI (HASSAS) SONUÇLARI VERMESİ

Genellikle mühendislik problemlerinin çözümü iki şekilde gerçekleştirilmektedir. Bunlardan birincisi direkt yöntem olup sonuç doğrudan elde edilir. Diğeri dolaylı çözüm olup sonuca iteratif (ardışık, tekrarlı) hesap yoluyla varılır. Dolaylı çözümlerde başlangıçta kabul edilen tolerans kadar hata yapılacağı açıktır. Durum böyle olmakla beraber problemlerin direkt çözümlerinde yuvarlatma hatalarının sonuçlara etkisi büyüktür. Dolaylı çözümlerde yuvarlatma hatalarının sonuçlara etkisi toleranas değeri gerektiği kadar küçük seçilirse gözardı edilecek durumdadır. Ayrıca değişken türlerinin duyarlıkları yükseltilerek, tamsayı yerine gerçel sayı, gerçel sayı yerine çift duyarlıklı sayı kullanılarak hassasiyet artırılabilir. Özellikle doğrusal (lineer) denklem sistemlerinin çözümlerinde pivotlama işlemi yapılarak doğruluk artırılabilir.

Program satırlarındaki zincirleme çarpma ve bölme işlemlerinde değişkenlerin sayısal büyüklükleri hakkında bilgi varsa;

Örnek: a ve b c' den çok büyük ise

d = $\frac{c}{b * a}$ işlemi ;

10 d = c/(b*a) yerine 10 d = c / b /a

şeklinde yapılırsa kısmen doğruluk artırılabildiği gibi kısmen de değişkenlerin ekstrem değerler almaları halinde olabilecek alt ve üst taşmaların önüne geçilebilir.

KESİN ve YAKLAŞIK SAYISAL DEĞERLER

Günlük yaşantımızda karşımıza çıkan sayısal değerler, kesin ve yaklaşık değerler olarak iki grupta toplanabilir. Çoğu durumlarda kesin (gerçek) değeri belirleme olanağı yoktur. Örneğin bir üçgenin iç açıları toplamı 180 derecedir dendiğinde buradaki 180 derece değeri kesin bir değerdir. Bir sınıfta 40 öğrenci var denildiğinde, bu 40 değeri de kesin bir sayıdır. Bir uzunluğun ölçülmüş değeri 10.284m denildiğinde ise buradaki 10.284m değeri yaklaşık bir sayısal değerdir. Çünkü ölçülerin her zaman rastgele ölçü hatalarıyla yüklü oldukları bir gerçektir. Aynı uzunluğu aynı ölçme aletiyle bir başka kişi yapsa büyük olasılıkla farklı bir değer elde

edecektir. Buradan ölçme yaparak gerçek değerin elde edilemeyeceği, ancak sayılabilen büyüklüklerin gerçek değeri elde edilebileceği ortaya çıkmaktadır. Ancak sayılan kitlenin çok büyük olması durumunda da hatasız değer elde etme zorlaşmaktadır. Örneğin ülke nüfusunun, nüfus sayım sonuçlarından hiç bir zaman tam olarak belirlenememesi gibi.

3. HESAPLAMALARA İLİŞKİN GENEL BİLGİLER

3.1 SAYILARIN YUVARLATILMASI

Hesaplamalar sırasında çarpma ve bölme işlemlerinde çıkan sonuçlar genellikle istenen sayıdaki basamaktan fazla olmaktadır. Bu sonuçların istenen basamak sayısında olması için sayıların yuvarlatılması gerekir. Yuvarlatma işlemi genel olarak şöyle yapılır.

Ondalık noktadan sonra (n) basamağı olan bir sayının (n-1) basamaklı olacak şekilde (bir basamak) yuvarlatılması klasik matematikte ve mühendislik bilimlerinde farklıdır.

Mühendislik bilimlerinde ;

(n). basamak > 5 ise (n). basamak atılır, (n-1). basamak bir artırılır.

(n). basamak < 5 ise sadece (n). basamak atılır.

(n). basamak = 5 ise (n). basamak atılır. Sistematik hata olmaması için yalnız (n-1). basamağın tek sayı olması durumunda (n-1). basamak bir artırılır.

Örnek : Aşağıdaki sayıları bir basamak yuvarlatınız.

$16.7654 \rightarrow 16.765$ $16.77 \rightarrow 16.8$

$218.92\mathbf{5} \rightarrow 218.92$, $218.93\mathbf{5} \rightarrow 218.94$

Klasik matematikte;

(n). basamak $\geq$ 5 ise (n). basamak atılır ve (n-1). basamak bir artırılır.

(n). basamak < 5 ise sadece (n). basamak atılır.

Örnek : Aşağıdaki sayıların bir basamak yuvarlatılmaları gösteriliyor.

$16.7654 \rightarrow 16.765$ $218.925 \rightarrow 218.93$ $16.77 \rightarrow 16.8$

Günümüz modern hesaplama araçları (hesap makineleri, bilgisayarlar) bu klasik matematik mantığına göre yuvarlatma yapmaktadırlar.

Aşağıda verilen sayıların her defasında birer basamaklarının yuvarlatılmaları gösteriliyor.

$16.7654 \rightarrow 16.765 \rightarrow 16.77 \rightarrow 16.8 \rightarrow 17$

$0.345001 \rightarrow 0.345 \rightarrow 0.35 \rightarrow 0.4$

$6.72504 \rightarrow 6.725 \rightarrow 6.73 \rightarrow 6.8 \rightarrow 7$

Eğer yuvarlatma işlemiyle birden çok sayıda basamağın yuvarlatılması isteniyorsa her defasında birer basamağın yuvarlatılmasına gitmek doğru olmaz. Yuvarlatma işleminin toptan bir kerede yapılması gerekir.

Bunun için ondalık noktadan sonra p basamaklı bir sayı ondalık noktadan sonra n basamaklı olacak şekilde yuvarlatılması isteniyorsa (n<p) diğer bir deyişle sayının sonundaki (p-n) basamak gösterilmek istenmiyorsa , Bu durumda; sayının sonundaki (p-n) basamağın oluşturduğu sayı $\geq 5 * 10^{(p-n-1)}$ ise n. basamaktaki rakam 1 artırılır değilse sadece sonraki (p-n) basamak atılır.

Örnek: 142.35489 sayısı ondalık noktadan sonra iki basamaklı olacak şekilde yuvarlatılmak isteniyor

Örneğimizde p = 5 ve n = 2 dir. Sayının yuvarlatılması istenen son üç basamağı 489 'dur.

$489 < 5*10^{(p-n-1)}$ yani 489<500 olduğu için son üç basamaktaki 489 atılır.

O halde 142.35489 $\rightarrow$ **142.35** olacak şekilde yuvarlatılır.

Oysa 142.35489 sayısını her defasında birer basamak yuvarlatarak ondalık noktadan sonra iki basamaklı hale getirseydik

142.35489 $\rightarrow$ 142.3549$\rightarrow$142.355$\rightarrow$**142.36**

yukarıdaki gibi 142.36 değerini elde ederiz. Ancak sayıların bu şekilde yuvarlatılması daha fazla duyarlık kaybına neden olacağından yanlıştır.

3.2 SAYILARDA ANLAMLI BASAMAK

Verilen sayının (ondalık noktadan sonra n basamaklı) doğruluğu hakkında bir bilgi yoksa ya da yuvarlatılmış bir sayı ise, sayının son basamağı, son basamak biriminin yarısı kadar $\pm 5*10^{-(n+1)}$ hatalı olabilir. Onun için verilen sayının son basamağına kuşku ile bakılabilir.

Örneğin; 216.84 olarak verilen bir sayı, gerçekte 216.835 ile 216.844 arasında olabilir.

Anlamlı basamak sayısı; sayının soldan, sıfırdan farklı ilk rakamından başlayarak sağa doğru sayılan rakamların sayısına denir.

$0.00275 \rightarrow$ 3 anlamlı basamak

$0.12752 \rightarrow$ 5 anlamlı basamak

$212.875 \rightarrow$ 6 anlamlı basamak

$47 * 10^2 \rightarrow$ 2 anlamlı basamak

$2501 \rightarrow$ 4 anlamlı basamak

Anlamlı basamak sayısı çarpma ve bölme işlemlerinde önemlidir. Çarpma ve bölme işlemlerinde, sonuç sayısının anlamlı basamak sayısı, işleme giren en küçük anlamlı basamağa sahip sayınınki kadar olmalıdır.

Örnek: $4.9178 * 2.03 = 9.983134 \rightarrow 9.98$

$712 * 0.21289 = 151.57768 \rightarrow 152$

$2.738 / 0.78 = 3.51025641 \rightarrow 3.5$

4. HESAPLAMALARDA HATA KAYNAKLARI

Sayısal yöntemlerle bir problemin çözümünde ortaya çıkan çeşitli hatalar nedeniyle, hesaplanan bir X değeri olması gereken gerçek değer x ' e genellikle eşit değildir, aralarında dx kadar bir fark vardır. Dikkatsizlikten kaynaklanan hesap hataları bu konunun dışındadır.

$$dx = x - X$$

Bu dx farkına Mutlak hata, Mutlak hatanın x'e ya da X'e oranına

$$Q_x = \frac{dx}{x} \approx \frac{dx}{X}$$

Bağıl hata diyoruz.

Hesaplamalarda karşılaşılan hata kaynaklarını üç başlık halinde özetleyebiliriz.

1- Yöntem Hataları
2- Giriş Verilerinin Hataları
3- Hesap (Yuvarlatma) Hataları

4.1. YÖNTEM HATALARI

Hesaplamada kullanılan yöntemin kesin değil de yaklaşık olmasından kaynaklanan hatalardır. Örneğin ardışık (iteratif) hesaplamayla cebirsel bir polinomun köklerinin bulunmasında sonsuz sayıda iterasyon, bir açının trigonometrik değerlerinin seri açınımıyla hesabında da sonsuz sayıda terim kullanılamayacağından sonuçlar her zaman bu tip hatayla yüklü olur. Bu hatalara kesme hatası da (truncation error) denir.

4.2. GİRİŞ VERİLERİNİN HATALARI

Hesaplamaya giren veriler daha başlangıçta bir takım hatalarla yüklüyse, hesaplama sonucunun da bu hatalarla yüklü olacağı kaçınılmazdır. Bu tür hataların yok edilmeleri olanaksızdır. Ancak giriş verilerindeki hataların sonuca etkisi kestirilebilir.

$y = f(x_1,x_2,....,x_n)$

biçimindeki bir fonksiyonda

$x_1, \quad x_2, \ldots, \quad x_n$ giriş verileri,

$\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ giriş verilerinin hataları olsun

Bu durumda hesaplanacak fonksiyon değeri (a_y) hatasıyla beraber

$$a_y+y = f(x_1+\Delta x_1 , x_2+\Delta x_2 ,...., x_n+\Delta x_n)$$

şekline dönüşür.

$y = f(x_1,x_2,....,x_n)$ fonksiyonunda Δx_i artımlarının sonuca etkisi , fonksiyonun tam diferansiyeli alınarak belirlenir.

$$\delta y = y'.\, dy$$

$$\delta y = \frac{dy}{dx_1}\,\Delta x_1 + \frac{dy}{dx_2}\,\Delta x_2 + \ldots + \frac{dy}{dx_n}\,\Delta x_n$$

δy hatasının maximum alabileceği değer artımların farklı işaretli olması durumu göz önünde bulundurulursa yukarıdaki eşitlikteki terimlerin mutlak değerleri alınarak aşağıdaki gibi hesaplanır.

$$a_y \le |\delta y| = \left|\frac{dy}{dx_1}\,(\Delta x_1)\right| + \left|\frac{dy}{dx_2}\,(\Delta x_2)\right| + \ldots + \left|\frac{dy}{dx_n}\,(\Delta x_n)\right|$$

Örnek:

$$y = \frac{x_1}{x_2}$$

şeklindeki bir fonksiyonda x_1 , x_2 üzerindeki Δx_1 ve Δx_2 artımlarının sonuca etkisi

$$\delta y = \frac{1}{x_2}\Delta x_1 - \frac{x_1}{x_2^2}\,\Delta x_2$$

şeklinde tam diferansiyel alınarak hesaplanır. Δx_i artımlarının değişik işaretli olması durumunda sonuca olacak maximum etkisi ,

$$a_y \le |\delta y| = \left|\frac{1}{x_2}\Delta x_1\right| + \left|-\frac{x_1}{x_2^2}\,\Delta x_2\right|$$

ise yukarıdaki gibi hesaplanır. Sayısal örnek olarak, $x_1 = 25$ ve $x_2 = 67$ olsun. y fonksiyonunun bu verilenlerle değeri ,

y = $\frac{x_1}{x_2} = \frac{25}{67}$ = 0.373134328 dir.

x_1 , x_2 üzerindeki Δx_1 = + 0.03 ve Δx_2 = +0.05 artımlarının sonuca etkisi, y fonksiyonunun tam diferansiyeli alınırsa,

$$\delta y = \frac{1}{x_2}\Delta x_1 - \frac{x_1}{x_2^2}\Delta x_2 = \frac{1}{67}(0.03) - \frac{25}{67^2}(0.05) = 0.000169302 \text{ dir.}$$

artımlar sonucu y fonksiyonunun gerçek değeri (y_g)

$y_g = \frac{25+0.03}{67+0.05}$ = 0.373303504 dır. Fark değeri,

$y_g - y$ = 0.373303504 - 0.373134328 = 0.000169176 dır. Görüldüğü gibi bu değer fonksiyonun tam diferansiyeli alınarak bulunan δy = 0.000169302' ye çok yakındır.

Δx_i artımlarının değişik işaretli olması durumunda sonuca olacak maximum etki ise ,

$$a_y \le |\delta y| = |\frac{1}{67}(0.03)| + |-\frac{25}{67^2}(0.05)| = 0.000726219\text{' dur.}$$

Örnek: Dikdörtgen şeklindeki bir parselin kenarları, a=100m ve b=350m olarak veriliyor. Kısa kenarın 2cm kısalması, uzun kenarın 3cm uzaması sonucu hesaplanacak alan ne kadar farklı olacaktır?

F=a*b=35000m^2 ; parselin ilk alanı

F_g=(100-0.02)*(350+0.03)=99.98*350.03=34995.9994m^2 ; parselin son alanı

$\Delta = F_g - F$=34995.9994 – 35000 = -4.0006$m^2 \cong$ -4m^2

veya F alan fonksiyonunun tam diferansiyelini alırsak bulunacak alan farkı

dF=b*da+a*db=350*(-0.02)+100*0.03= -7+3= -4m^2 olur.

Giriş verilerindeki artımların hesap sonucuna etkisi, şayet artımların işareti biliniyorsa fonksiyonunun diferansiyeli alınarak belirlenebilmektedir. Gerçi bu durumda artımların işareti bilindiğine göre düzeltilmiş verilerle fonksiyonun gerçek değeri de hesaplanabilir. Oysa, pratikte giriş verileri genellikle çeşitli deney ve gözlem sonucu ölçümlerle elde edilir. Dolayısı ile ölçümlerdeki artımların (hataların) işareti bilinmez ve ölçü hatalarının da daima ± işaretli olacağı göz önünde tutulursa ölçülerin fonksiyonunun hatasının, hataların yayılma kuralına göre fonksiyonun *karesel ortalama hatası* alınarak belirlenmesi hata teorisi açısından daha uygun olur. Şimdi bir fonksiyonun karesel ortalama hatasının nasıl bulunacağını görelim.

$y = f(x_1, x_2, ..., x_n)$

şeklindeki bir fonksiyonun ortalama hatası Hataların Yayılma Kuralına göre

x_1, x_2, ..., x_n giriş verilerini (veriler korelasyonsuz)

$\pm m_1$, $\pm m_2$,...., $\pm m_n$ giriş verilerinin karesel ortalama hataları(standart sapmaları)

olsun

y sonucunun ortalama hatası (karesel ortalama hatası) m_y

$$m_y^2 = \left(\frac{dy}{dx_1}\right)^2 m_1^2 + \left(\frac{dy}{dx_2}\right)^2 m_2^2 + \ldots\ldots\ldots + \left(\frac{dy}{dx_n}\right)^2 m_n^2 = \sum_{i=1}^{n}\left(\frac{dy}{dx_i} m_i\right)^2$$

şeklinde hesaplanmaktadır.

Örnek:

$$y = \frac{x_1}{x_2}$$

şeklindeki bir fonksiyonun karesel ortalama hatası

$$m_y^2 = \left(\frac{1}{x_2}\right)^2 m_1^2 + \left(\frac{-x_1}{x_2^2}\right)^2 m_2^2$$

olmaktadır. Sayısal örnek olarak, $x_1 = 25$ ve $x_2 = 67$ ve hatalar sırasıyla $m_1 = \pm 0.03$ ve $m_2 = \pm 0.05$ olsun. y fonksiyonunun bu verilenlerle karesel ortalama hatası ,

$$m_y^2 = \left(\frac{1}{x_2}\right)^2 m_1^2 + \left(\frac{-x_1}{x_2^2}\right)^2 m_2^2 = \left(\frac{1}{67} 0.03\right)^2 + \left(\frac{-25}{67^2} 0.05\right)^2 = 2.78E\text{-}7$$

$m_y = \pm 0.000527$ olur.

4.3. YUVARLATMA HATALARI

Hesaplayıcıların sonlu sayıda basamak uzunluğuyla hesap yapmasından kaynaklanmaktadır. Bu nedenle hesaplayıcının bize verdiği değer genellikle yuvarlatılmış bir sonuçtur. Yuvarlatma hataları, hesaplamalarda kullanılan basamak sayısı bir iki basamak artırılarak ya da genel olarak çift duyarlıklı sayılarla çalışarak azaltılabilir.

Hesaplama Kuralları:

1) Kural olarak hesaplamalarda yuvarlatma hatalarının sonuca etkisinin ölçülerde ulaşılan duyarlığın onda birini geçmemesi istenir. Diğer bir deyişle yuvarlatma hataları hesapla bulunacak değerler için ayrıca bir hata kaynağı olmamalıdır. Bu nedenle hesaplamalarda ara işlemlerde yuvarlatma yapılmaz. Bunun için gerekirse hesaplayıcının hafızalarından yararlanılır.

2) Bir sayısal hesaplamanın doğruluğu, işleme giren parametrelerin doğruluğuna bağlıdır. Hesaplama sonucunu ondalık noktadan sonraki basamak sayısını çoğaltarak ifade etmek, sonucun doğruluğunu artırmaz. Hesap sonucu bulunan değer, hiçbir zaman verilenlerden daha duyarlı olamaz. Bunun için sonuç değerler anlamlı olacak şekilde yuvarlatılır.

Söz gelişi bir uzunluk arazide üç kez ölçülmüş 142.36m , 142.37m ve 142.34m değerleri elde edilmiştir. Bu uzunluğun kesin değerini elde etmek için verilen üç ölçünün aşağıdaki gibi aritmetik ortalaması alınır.

$$x = \frac{142.36 + 142.37 + 142.34}{3} = 142.356666667\text{m}$$

Bu işlem hesap makinesi ile yapıldığında yukarıdaki değer elde edilir. Ancak x kesin değerini bu şekilde ifade etmek doğru değildir. Çünkü x =142.356666667m. ifadesi 10^{-9}m. duyarlık ifade eder. Oysa yapılan ölçüler cm. duyarlığındadır. Bu nedenle kesin değer anlamlı olacak şekilde yuvarlatılarak x =142.357m ya da x =142.36m olarak ifade edilir.

Benzer şekilde bir büyüklüğün ölçülmüş değerinin sayısal olarak 25 ya da 25.00 olarak ifade edilmesi cebrik olarak aynı anlama gelmekle beraber ölçme tekniği açısından farklı anlam taşır ve ikinci yazım biçimi ölçmenin yüzde bir duyarlıkta yapıldığını gösterir.

Hesaplama için birden fazla yöntem olması durumunda bunlardan direkt olarak ham verileri kabul eden ve ara işlemlere olabildiğince gereksinim duymayan yöntemler hesaplama tekniği açısından yeğlenmelidir. Hangi yöntemin daha duyarlı olduğu diferansiyel bağıntılar çıkarılarak belirlenebilir.

SERİLER İLE HESAPLAMALAR

Kesin değerleri belirlenemeyen bazı büyüklükler örneğin Π ve e sayıları gibi trigonometrik ve logaritmik fonksiyonlar da seri açınımlarından yararlanarak hesaplanabilmektedir. Elektronik hesaplayıcılarda bu fonksiyon değerlerini seri açınımlarından yeterli doğruluk verecek sayıda terim kullanarak hesaplamaktadır. Bir f(x) fonksiyonu x_o yaklaşık değerinden yararlanarak Taylor serisine açıldığında,

$$x = x_o + \Delta x$$

$$f(x_o+\Delta x) = f(x_o) + \frac{f'(x_o)}{1!}\Delta x + \frac{f''(x_o)}{2!}\Delta x^2 + \frac{f'''(x_o)}{3!}\Delta x^3 + \ldots\ldots + \frac{f^{(n)}(x_o)}{n!}\Delta x^n$$

olur. Yukarıdaki Taylor serisinde $x_o = 0$ alınırsa Mac Laurin serisi elde edilir.

$$f(\Delta x) = f(0) + \frac{f'(0)}{1!}\Delta x + \frac{f''(0)}{2!}\Delta x^2 + \frac{f'''(0)}{3!}\Delta x^3 + \ldots\ldots + \frac{f^{(n)}(0)}{n!}\Delta x^n$$

Burada $f', f'', \ldots, f^{(n)}$ gösterimleri f(x) fonksiyonunun sırayla x değişkenine göre birinci, ikinci ve (n). türevlerini göstermektedir. Bu şekilde seri açınımları yapılırsa trigonometrik,logaritmik fonksiyonlar ve bunların tersleri aşağıdaki gibi Mac Laurin serilerinden hesaplanabilir.

Fonksiyon	Seri Açınımı	Geçerli Aralık
Sin x =	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots\ldots$	x radyan $(-\infty<x<+\infty)$
Cos x =	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots\ldots$	x radyan $(-\infty<x<+\infty)$
Tan x =	$x + \frac{x^3}{3} + \frac{2x^5}{15} + \frac{17x^7}{315} + \ldots\ldots$	x radyan $(\lvert x\rvert<\Pi/2)$

Cot x =	$\frac{1}{x} - \frac{x}{3} - \frac{x^3}{45} - \frac{-2x^5}{945} - \ldots\ldots$	x radyan (0<|x|<Π)
Arcsin x =	$x + \frac{1.x^3}{2.3} + \frac{1.3.x^5}{2.4.5} + \ldots\ldots$	(-1=<x=<+1)
Arctan x =	$x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots\ldots$	(-∝<x<+∝)
e =	$1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots\ldots$	
e^x =	$1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots$	
ln(1+x) =	$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots\ldots$	|x|<1
Π =	$4*(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\ldots)$	

5. M A T R İ S L E R

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} \end{bmatrix}$$

Matris, satır ve sütun düzeninde yerleştirilmiş düzgün dikdörtgen biçimli sayılar topluluğudur. m satır ve n sütunlu bir A matrisinin mxn elemanı vardır. Matrisler genellikle büyük harflerle gösterilirler. Matrislerin, tek bir değişken adı altında çok sayıda değişmezin adreslenebilmesine olanak sağlaması bilgisayar hesaplamalarında çok büyük kolaylıklar sağlamaktadır. Örneğin yukarıdaki A matrisinde mxn adet sayı depolanmaktadır.

Yukarıdaki a_{ij}'ler A matrisinin elemanlarıdır. Genel olarak a_{ij} ile gösterilen elemanların birinci indisi olan i değeri elemanın bulunduğu satır sıra sayısını, ikinci indis j ise sütun sıra sayısını göstermektedir.

Örnek Program : MATRİS YAZDIRMA

Bu programı bir $A_{m,n}$ matrisini data satırlarından okuyup yazmaktadır. Matrisin boyutları m=3 ve n=2 110 ve 120 nolu deyimlerde verilmiştir.

```
100 REM Bu program A(mn) matrisinin okur ve yazdırır
102 CLS :rem
103 PRINT , " Bu program A(mn) matrisini okur ve yazar"
110 m = 3: PRINT , " matrisin satır sayısı   m ="; m
120 n = 2: PRINT , " matrisin sütun sayısı n="; n
125 DIM a(m, n)
150 FOR i = 1 TO m: FOR j = 1 TO n: READ a(i, j): NEXT j:
NEXT i
160 PRINT , " A  Matrisi"
170 FOR i = 1 TO m: FOR j = 1 TO n: PRINT , a(i, j); : NEXT j:
PRINT , : NEXT i
200 DATA 3,5,7,1,2,4
```

PROGRAM ÇIKTISI

```
Bu program A(mn) matrisini okur ve yazar
 matrisin satır sayısı   m = 3
 matrisin sütun sayısı n= 2
 A Matrisi
 3  5
 7  1
 2  4
```

ÖZEL MATRİS TÜRLERİ

m satır n sütunlu bir $A_{m,n}$ matrisinde;

m = n ise A matrisine kare matris

$m \neq n$ ise A matrisine dikdörtgen matris

m > n ise A matrisine dik matris

m < n ise A matrisine yatık matris

denilmektedir.

VEKTÖRLER

Bir satır ya da bir sütundan oluşan matrislere vektör denilmektedir.Vektörler genelde küçük harflerle gösterilir.

*** Sütun Vektörü**

$$\mathbf{b_{m,1}} = \mathbf{b_m} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_m \end{bmatrix}$$

*** Satır Vektörü**

$$\mathbf{c_{1,n}} = \mathbf{c_n} = \begin{bmatrix} c_1 & c_2 & c_3 & \dots\dots & c_n \end{bmatrix}$$

*** Bir Vektörü**

Bütün elemanları bir (1) olan sütun vektörüdür.

$$\mathbf{e_m} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

*** Birim Vektör**

Elemanlarından yalnız bir tanesi bir (1) diğerleri sıfır (0) olan özel bir sütun vektörüdür.

$$\mathbf{e_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \qquad \mathbf{e_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \dots \\ 0 \end{bmatrix}$$

yukarıdaki soldaki birim vektörün sadece 2. elemanı bir (1) olduğu için e_2 olarak adlandırılmaktadır.

*** Sıfır Vektörü**

Bütün elemanları sıfır (0) olan bir sütun vektörüdür.

$$\mathbf{0_m} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

*** Köşegen Matris**

Ana köşegendeki elemanları hariç diğer bütün elemanları sıfır olan özel bir kare matristir.

$$D = \begin{bmatrix} d_{11} & 0 & 0 & \dots & 0 \\ 0 & d_{22} & 0 & \dots & 0 \\ 0 & 0 & d_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & d_{nn} \end{bmatrix} \qquad d_{ij} = 0 \quad (i \neq j)$$

*** Skalar Matris**

Köşegen elemanları birbirine eşit olan özel bir köşegen matristir.

$$A = \begin{bmatrix} a & 0 & 0 & \dots & 0 \\ 0 & a & 0 & \dots & 0 \\ 0 & 0 & a & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a \end{bmatrix} \qquad \begin{aligned} a_{ij} &= a \quad (i = j) \\ a_{ij} &= 0 \quad (i \neq j) \end{aligned}$$

*** Sıfır Matris**

Bütün elemanları sıfır (0) olan bir matristir.

$$A = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

*** Birim Matris**

Köşegen üzerindeki elemanları 1 diğer tüm elemanları sıfır olan özel bir skalar matristir. Genellikle I ya da E harfiyle gösterilir.

$$I = E = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$e_{ij} = 1 \quad (i = j)$$

$$e_{ij} = 0 \quad (i \neq j)$$

*** Transpoz (Evrik) Matris**

Bir matrisin satırları ile sütunlarının yer değiştirilmesi sonucunda elde edilen matristir $(.)^T$ ya da (.)‘ üst indisi ile gösterilir.

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 2 \\ 4 & 7 \\ 5 & 1 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 7 & 1 \end{bmatrix}$$

Örnek Program : TRANSPOZ MATRİS

Bu programı bir $A_{m,n}$ matrisini data satırlarından okuyup transpozunu hesaplamaktadır. Matrisin boyutları m=3 ve n=2 110 ve 120 nolu deyimlerde verilmiştir.

```
100 REM Bu program A(mn) matrisinin transpozu AT(nm)hesaplar
ve yazar
110 m = 3: PRINT , "matrisin satır sayısı  m ="; m
120 n = 2: PRINT , "matrisin sütun sayısı n="; n
125 DIM a(m, n), at(n, m)
150 FOR i = 1 TO m: FOR j = 1 TO n: READ a(i, j): NEXT j:
NEXT i
160 PRINT , " A  Matrisi"
170 FOR i = 1 TO m: FOR j = 1 TO n: PRINT , a(i, j); : NEXT j:
PRINT , : NEXT i
180 FOR i = 1 TO m: FOR j = 1 TO n: at(j, i) = a(i, j): NEXT
j: NEXT i
185 PRINT , " AT  Transpoz  Matrisi"
190 FOR i = 1 TO n: FOR j = 1 TO m: PRINT , at(i, j); : NEXT
j: PRINT , : NEXT i
200 DATA 3,5,7,1,2,4
```

PROGRAM ÇIKTISI

```
matrisin satır sayısı  m = 3
matrisin sütun sayısı n= 2

 A Matrisi
 3  5
 7  1
 2  4
 AT  Transpoz  Matrisi
 3  7  2
 5  1  4
```

*** Transpoz (Evrik) Vektör**

Bir satır vektörünün sütun vektör ya da bir sütun vektörünün satır vektör biçiminde yazılmış halidir.

$$b = \begin{bmatrix} 1 & 2 & 5 & 3 & 4 \end{bmatrix} \qquad b^T = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 3 \\ 4 \end{bmatrix}$$

Bir matrisin ya da bir vektörün transpozunun transpozu kendisine eşittir.

$(A^T)^T = A \qquad (b^T)^T = b$

skalar bir sayının transpozu kendisine eşittir.

*** Simetrik Matris**

Ana köşegenine göre simetrik elemanları eşit olan bir kare matristir. Simetrik matrislerin transpozları kendisine eşittir.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{12} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{13} & a_{23} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & a_{3n} & \dots & a_{nn} \end{bmatrix} \qquad (a_{ij} = a_{ji})$$

$$A^T = A$$

Örnek:

$$A = \begin{bmatrix} 3 & 4 & 2 \\ 4 & 6 & 7 \\ 2 & 7 & 9 \end{bmatrix}$$

*** Ters Simetrik Matris (Antisimetrik, Skewsimetrik Matris)**

Ana köşegenindeki bütün elemanları sıfır, köşegene göre simetrik elemanları zıt işaretli olan özel bir simetrik matristir.

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

$$a_{ij} = -a_{ji} \quad (i \neq j)$$
$$a_{ij} = 0 \quad (i = j)$$

Bir antisimetrik matrisin transpozu kendisinin ters işaretlisidir.

$$A^T = -A$$

Her kare matris, biri simetrik diğeri antisimetrik olmak üzere toplamları kendini verecek şekilde iki matrise ayrılabilir. Başlangıç matrisimiz C olsun, ayırma işlemi sonunda simetrik matris A , Antisimetrik matris B olsun A ve B matrisinin elemanları

$$a_{ij} = \tfrac{1}{2}(c_{ij}+c_{ji}) \qquad b_{ij} = \tfrac{1}{2}(c_{ij}-c_{ji})$$

eşitliklerinden hesaplanır.

Örnek:

$$C = \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

olsun C = A + B olacak şekilde A ve B matrisinin elemanlarını hesaplayalım.

$a_{11} = \tfrac{1}{2}(2+2) = 2$ $\quad b_{11} = \tfrac{1}{2}(2-2) = 0$
$a_{12} = \tfrac{1}{2}(3+5) = 4$ $\quad b_{12} = \tfrac{1}{2}(3-5) = -1$
$a_{21} = \tfrac{1}{2}(5+3) = 4$ $\quad b_{21} = \tfrac{1}{2}(5-3) = 1$
$a_{22} = \tfrac{1}{2}(6+6) = 6$ $\quad b_{22} = \tfrac{1}{2}(6-6) = 0$

$$A = \begin{bmatrix} 2 & 4 \\ 4 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Simetrik Antisimetrik

$$\underset{A}{\begin{bmatrix} 2 & 4 \\ 4 & 6 \end{bmatrix}} + \underset{B}{\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}}$$

*** Üçgen Matris**

Ana köşegenin altındaki ya da üstündeki elemanları sıfır olan özel bir kare matristir.

*** Üst Üçgen Matris:**

Ana köşegenin altındaki elemanları sıfır olan özel bir kare matristir.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

$$a_{ij} = 0 \quad (i>j)$$

*** Alt Üçgen Matris:**

Ana köşegenin üstündeki elemanları sıfır olan özel bir kare matristir.

$$B = \begin{bmatrix} b_{11} & 0 & 0 & \dots & 0 \\ b_{21} & b_{22} & 0 & \dots & 0 \\ b_{31} & b_{32} & b_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{bmatrix} \qquad b_{ij} = 0 \quad (i<j)$$

*** Band Matris**

Ana köşegeni ve buna paralel bir kaç alt ya da üst köşegendeki elemanların dışında bütün elemanları sıfır olan matristir.

$$A = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 \\ 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 2 & 4 & 5 \\ 0 & 0 & 0 & 1 & 3 \end{bmatrix}$$

*** Çok Boşluklu (Sparse) Matris**

Elemanlarının pek çoğu sıfır olan ve dolu elemanların dizilişi genelde herhangi bir sistematik göstermeyen matrislerdir.

*** Kofaktör Matris**

Kofaktör matrisi tanımlamadan önce, minör ve kofaktör deyimlerini tanımlayalım. Bir A kare matrisindeki herhangi bir a_{ij} elemanının bulunduğu satır ve sütun silinir, boyutu bir küçülmüş olan matrisin determinantı M_{ij} nin $(-1)^{i+j}$ ile çarpıldıktan sonra bulunan büyüklüğü A_{ij} ile gösterelim. M_{ij} 'ye a_{ij} elemanının minörü, A_{ij} ' ye de kofaktörü denir.

$$\text{Kofaktör } A_{ij} = (-1)^{i+j}.\, M_{ij}$$

Bir A kare matrisinin her a_{ij} elemanı yerine o elemanların kofaktörlerini yazarsak A matrisinin kofaktör biçimini elde eder ve COF-A ile gösteririz.

$$\text{COF-A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2n} \\ A_{31} & A_{32} & A_{33} & \dots & A_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & A_{n3} & \dots & A_{nn} \end{bmatrix}$$

*** Ek (Adjoint) Matris**

Kofaktör matrisin transpozuna Ek Matris denir ve **Adj.A** ile gösterilir.

$$\text{Adj.A} = \begin{bmatrix} A_{11} & A_{21} & A_{31} & \dots & A_{n1} \\ A_{12} & A_{22} & A_{32} & \dots & A_{n2} \\ A_{13} & A_{23} & A_{33} & \dots & A_{n3} \\ \dots & \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & A_{3n} & \dots & A_{nn} \end{bmatrix}$$

$$\text{Adj.A} = (\text{COF-A})^T$$

***Ortogonal Matris**

Bir A kare matrisi $A^T = A^{-1}$ koşulunu sağlıyor ise A matrisine ortogonal matris denir.

$A^TA = AA^T = E$ Ortogonallik koşulu

<u>Örnek:</u>

$$A = \begin{bmatrix} \sqrt{3}/2 & 1/2 \\ -1/2 & \sqrt{3}/2 \end{bmatrix}$$

şeklinde verilen A matrisinin tersi ve transpozu

$$A^T = A^{-1} = \begin{bmatrix} \sqrt{3}/2 & -1/2 \\ 1/2 & \sqrt{3}/2 \end{bmatrix}$$

dır. Yukarıda görüldüğü gibi A matrisinin tersi ve transpozu birbirlerine eşittir. Dolayısıyla A matrisi bir ortogonal matristir.

*** Periyodik Matris**

Bir A kare matrisi pozitif bir k tamsayısı için

$$A^{k+1} = A$$

koşulunu sağlıyor ise A matrisine periyodik matris, k pozitif tamsayısına da A matrisinin periyodu denir.

Özel olarak k = 1 için $A^2 = A$ olur. Bu durumda A matrisine idempotent matris denir.

Örnek:

$$A = \begin{bmatrix} 1 & -2 & -6 \\ -3 & +2 & 9 \\ 2 & 0 & -3 \end{bmatrix}$$

şeklinde verilen A matrisi, $A^3 = A$ koşulunu sağladığı için periyodik bir matristir. Burada A matrisinin periyodu k=2 dir.

*** Nilpotent Matris**

Bir A kare matrisi pozitif bir k tamsayısı için

$$A^k = 0$$

koşulunu sağlıyor ise A matrisine nilpotent matris denir.

Örnek:

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

şeklinde verilen A matrisi, $A^2 = 0$ koşulunu sağladığı için Nilpotent bir matristir.

***İnvolutif Matris**

Bir A kare matrisi

$$A = A^{-1}$$

koşulunu sağlıyor ise A matrisine İnvolutif matris denir.

***Pozitif Belirli Matris**

Simetrik bir A matrisi, sıfırdan farklı her x vektörü için

$$x^T Ax > 0$$

koşulunu sağlıyor ise A matrisine pozitif belirli (definit) matris

$$x^T Ax \geq 0$$

koşulunu sağlıyor ise A matrisine pozitif yarı belirli matris denir.

*** Alt Matris**

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & . & a_{14} \\ a_{21} & a_{22} & a_{23} & . & a_{24} \\ a_{31} & a_{32} & a_{33} & . & a_{34} \\ \dots & \dots & \dots & . & \dots \\ a_{41} & a_{42} & a_{43} & . & a_{44} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Herhangi bir matrisin bazı satırlarının ve / veya bazı sütunlarının silinmesiyle elde edilen daha küçük boyutlu matrislere denir. Yukarıda A matrisi dörde bölünerek A_{11}, A_{12}, A_{21} ve A_{22} alt matrisleri elde edilmiştir.