Bölüm 2

Bilgisayar Sayı Sistemi, Aritmetiği ve Hata

Sayısal yöntemler bilgisayarlarda uygulanır, ancak bilgisayarlar reel veya kompleks sayılar kümesi yerine, bu kümelerin sonlu sayıda elemandan oluşan *çok özel* alt kümeleri üzerinde çalışır. Bu bölümde öncelikle *bilgisayar sayı sistemi* adı verilen bu *çok özel* alt kümeyi ve bu kümenin özelliklerini inceleyeceğiz. Ayrıca bilgisayar sayı sistemi üzerindeki aritmetik işlemleri inceleyerek, bu işlemlerin reel sayılar kümesi üzerinde alışık olduğumuz kapalılık ve birleşme gibi elemanter özelliklere sahip olmadığını göreceğiz. Böylece gerek bilgisayar sayı sistemi ve gerekse sayı sistemi üzerindeki aritmetik işlemleri yakından tanıyarak, sayısal analiz sürecinde sayı sistemi ve aritmetiğinden kaynaklanabilecek hataları minimize etmenin yollarını öğrenmiş olacağız.

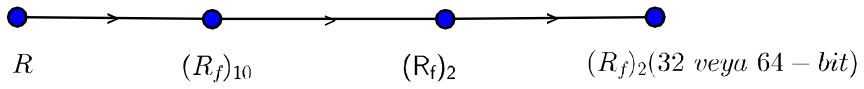
Bu amaçla bu bölümde sırasıyla

- hata kavramını inceleyerek,
- R ile gösterilen reel sayılar kümesinin bir alt kümesi olan ve $R_f(10)$ ile göstereceğimiz on tabanlı kayan nokta sayı sistemini,
- R den $R_f(10)$ sayı sistemine, yani,

$$R- > R_f(10)$$

  dönüşümü ve bu dönüşümde oluşabilen hataları,

- $R_f(2)$ ile göstereceğimiz iki tabanlı kayan nokta sayı sistemini,



Şekil 2.1: R den $R_f(2)$ ye dönüşüm süreci

- $R_f(10)$ dan $R_f(2)$ kayan nokta sistemine, yani,

$$R_f(10) -> R_f(2)$$

dönüşümü ve bu dönüşümde oluşan hataları,

- İki tabanlı kayan noktalı sayıların $32-$bit formata dönüşümü:

$$R_f(2) \rightarrow R_f(2)(32-bit)$$

veya $64-$bit formata dönüşümü:

$$R_f(2) \rightarrow R_f(2)(64-bit)$$

ve bu dönüşümlerde oluşabilen hataları inceliyoruz. Özetle şematik olarak Şekil 2.1 ile ifade edilen ve reel sayılar kümesinden bilgisayar sayı sistemine dönüşüm sürecinin her bir aşamasında oluşabilen hataları inceliyoruz.

- Ayrıca, bilgisayar sayı sisteminden kaynaklanan ve anlam kaybı hatası adı verilen özel bir hatanın nasıl oluştuğunu örnekler üzerinde gözlemliyoruz.

- Yukarıdaki her bir aşamada oluşması mümkün olan hatanın ilerleyen aritmetik işlemlere nasıl yayıldığının incelenmesi de ayrı bir konudur ve bu konuyu Taylor yaklaşımlarıyla ilgili olduğu için bir sonraki bölümün son kısmında inceliyoruz.

Konuyla ilgili detaylar için, bu bölümü hazırlarken faydalandığımız [1],[3], [4],[7],[8] ve [10] temel kaynaklarını öneriyoruz.

2.1 Hata(kesme ve yuvarlama hataları)

Ölçüm işlemlerinde genelde gerçek değer yerine, gerçek değeri en yakın biçimde temsil eden yaklaşımı kullanırız. Bu durumda

$$\textit{hata=gerçek değer-yaklaşım}$$

bağıntısı ile tanımlanan ve yaklaşımdan kaynaklanan bir hata oluşur ki bu hata sayısal analizde mutlak hata olarak tanımlanır.

TANIM 2.1. *x ile gösterilen bir büyüklüğün gerçek değeri ile bu değeri temsil eden x_f değeri arasındaki fark x_f yaklaşımının* ***mutlak hatası*** *veya kısaca* ***mutlak hata*** *olarak tanımlanır ve*

$$\Delta x = x - x_f$$

notasyonu ile gösterilir.

x_f yaklaşım değeri, x gerçek değerinden aşağıda tanımlandığı üzere *kesme* veya *yuvarlama* yapmak suretiyle elde edilebilir:

TANIM 2.2. *$x = d_0.d_1d_2....d_nd_{n+1}...$ değerine, noktadan(virgülden) sonraki n basamakla* ***kesme*** *esasına göre elde edilen yaklaşım $x_f = d_0.d_1d_2....d_n$ olup, bu yaklaşım sonucu oluşan*

$$\Delta x = x - x_f = 0.\overbrace{0..^{n\ adet}..0}\, d_{n+1}\cdots$$

hatasına mutlak kesme(chopping) hatası adı verilir.

Örneğin $x = 1.76$[1] değerine noktadan sonra bir basamakla kesme esasına göre elde edilen yaklaşım $x_f = 1.7$ olup, bu yaklaşım sonucunda oluşan mutlak kesme hatası ise

$$\Delta x = x - x_f = 0.06$$

dır.

Güncel bilgisayarlar bu yaklaşım yöntemini kullanmamaktadır, dolayısıyla bu bölümde hata kavramı deyince aksi belirtilmedikçe teknik olarak aşağıda tanımlanan ve genelde daha küçük yaklaşım hatasına neden olan *yuvarlama hatasını* ifade ediyor olacağız.

[1]Kesirli sayılarda kullanılan nokta, virgül anlamındadır. Uygulamalarımızda kullandığımız MATLAB/Octave ile uyum açısından virgül yerine biz de nokta kullanıyoruz.

TANIM 2.3. *On tabanlı sistemde* $x = d_0.d_1d_2....d_nd_{n+1}(0 \leq d_i \leq 9)$ *değerine, noktadan sonraki* n *basamakla* ***en yakın sayıya yuvarlama*** *esasına göre elde edilen yaklaşım*

$$x_f = \begin{cases} d_0.d_1d_2....d_n & eğer\, diğer\ d_{n+1}ler \\ d_0.d_1d_2....d_n + (1/10)^n, & d_{n+1} \geq 5 \end{cases}$$

olup, bu yaklaşım sonucu oluşan

$$\Delta x = x - x_f$$

hatasına mutlak yuvarlama(round off) hatası adı verilir.

Yukarıdaki örnekte $x = 1.76$ değerine noktadan sonra bir basamakla yuvarlama esasına göre elde edilen yaklaşım

$$x_f = 1.7 + (1/10)^1 = 1.8$$

olup, bu yaklaşım sonucunda oluşan mutlak yuvarlama hatası ise -0.04 tür.

Öte yandan $x = 1.73$ değerine noktadan sonra bir basamakla yuvarlama esasına göre elde edilen yaklaşım

$$x_f = 1.7$$

olup, bu yaklaşım sonucunda oluşan mutlak yuvarlama hatası ise 0.03 tür.

Mutlak hata günlük hayatta algıladığımız hata kavramına tam olarak karşılık gelmez:

Örneğin $x = 10.1$ birim uzunluğa sahip olan cismin uzunluğu $x_f = 10$ birim olarak ölçülmüşse, x_f yaklaşımında oluşan mutlak hata veya mutlak yuvarlama hatası

$$\Delta x = x - x_f = 10.1 - 10 = 0.1$$

değerine eşittir.

Öte yandan $x = 1.1$ birim gerçek uzunluğuna sahip olan başka bir cismin uzunluğu $x_f = 1$ birim olarak ölçülmüşse, x_f yaklaşımının mutlak hatası

$$\Delta x = 1.1 - 1 = 0.1$$

değerine sahiptir.

Her iki ölçüm sonucunda oluşan mutlak hatalar birbirine eşit olmasına rağmen, her nedense ikinci ölçümde daha fazla hata yaptığımızı düşünürüz.

Dolayısıyla mutlak hata kavramının güncel hayatta *algıladığımız hata* kavramına tam olarak karşılık gelmediğini görüyoruz!

Bu durumda güncel algılarımıza uygun bir hata kavramı mevcut olmalıdır ki bu kavram aşağıdaki tanım ile verilmektedir.

TANIM 2.4. *$x \neq 0$ değerine $x_f \neq 0$ ile yuvarlama prensibine göre yaklaşımda oluşan mutlak hatanın x e oranı olarak tanımlanan hataya x_f yaklaşımının* ***bağıl yuvarlama hatası*** *veya kısaca* ***bağıl hatası*** *adı verilir ve*

$$\varepsilon_b(x) = \frac{\Delta x}{x} \doteq \frac{\Delta x}{x_f}$$

notasyonu ile gösterilir.

Buna göre yukarıdaki birinci ölçümde oluşan bağıl hata

$$\varepsilon_b(x) = \frac{\Delta x}{x} = \frac{0.1}{10.1} \doteq \frac{0.1}{10} = 0.01$$

ve ikincide oluşan hata ise

$$\varepsilon_b(x) = \frac{\Delta x}{x} = \frac{0.1}{1.1} \doteq \frac{0.1}{1} = 0.1$$

dir, yani ikinci ölçüm sonucu yapılan bağıl hata beklentilerimiz doğrultusunda daha büyüktür. O halde bağıl hata günlük hayatta algıladığımız hata ile daha uyumludur.

Ölçüm sonuçları için yapmak durumunda olduğumuz yaklaşımlara benzer olarak, bilgisayarların sınırlı bellek kapasiteleri ve hız limitleri nedeniyle de, sayısal analiz sürecinde bilgisayarlar üzerinde elimizdeki verilerle çalışmak yerine, onları temsil eden yaklaşımlarla çalışmak durumunda kalırız. Bu durumda kesme veya yuvarlama hataları adını verdiğimiz hatalar oluşur. Sayısal analiz sürecinde oluşabilecek olan kesme veya yuvarlama hatalarını anlayabilmek için öncelikle aşağıda tanımlanan $R_f(10)$,$R_f(2)$ sayı sistemleri ve $R_f(2)$ sayı sisteminin $R_f(2)-$32-bit ve $R_f(2)-$64-bit bellek gösterimlerini yakından incelemeliyiz.

2.2 $R_f(10)$ Kayan nokta sayı sistemi

Bu bölümde $R_f(10)$ sayı sistemini oluşturan sayıların özelliklerini ve $R- >$

$R_f(10)$ dönüşümünde oluşan mutlak ve bağıl kesme ve yuvarlama hatalarını inceliyoruz.

Sonlu sayıda ondalık basamağa sahip herhangi bir x reel sayısı, uygun $m \geq 0, n \geq 0$ tamsayıları için

$$U = \{10^n, 10^{n-1}, \cdots, 10^0, 10^{-1}, \cdots, 10^{-m}\} \tag{2.1}$$

kümesinin elemanlarının lineer bileşimi olarak ifade edilebilir. Diğer bir deyimle, uygun

$$c_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

için

$$\begin{aligned} x &= (c_1 c_2 \cdots c_{n+1}.d_1 \cdots d_{m-1} d_m)_{10} \\ &= c_1 10^n + c_2 10^{n-1} + \cdots + c_{n+1} 10^0 \\ &\quad + d_1 10^{-1} + \cdots d_{m-1} 10^{-m-1} + d_m 10^{-m} \end{aligned} \tag{2.2}$$

biçiminde ifade edilebilir. Burada $c_1 c_2 \cdots c_{n+1}$ sayının tam kısmını ve $.d_1 \cdots d_{m-1} d_m$ ise kesirli kısmını ifade eder.

x sayısının U kümesinin elemanlarının lineer bileşimi olarak bu şekildeki ifadesine x in onluk sisteme(on tabanlı sisteme) göre gösterimi adı verilmektedir. Örneğin

$$1964.24 = 1 \times 10^3 + 9 \times 10^2 + 6 \times 10^1 + 4 \times 10^0 + 2 \times 10^{-1} + 4 \times 10^{-2}$$

olarak $(n = 3, m = -2)$ ifade edilebilir. O halde verilen sayı,

$$U = \{10^3, 10^2, 10^1, 10^0, 10^{-1}, 10^{-2}\}$$

kümesini elemanları cinsiden(veya lineer cebirsel terminoloji ile taban elemanlarının lineer bileşimi olarak) ifade edilmiştir.

En genel olarak $R_f(10)$ sayı sistemi

$$x_f = \pm q_f \times 10^e$$

formatında ifade edilebilen sayılardan oluşur ve bu formatta ifade edilebilen sayılara *on tabanlı kayan noktalı sayılar* adı verilir, burada q_f ye, x_f nin **kesir kısmı(veya mantis)** adı verilir ve

$$q_f = 0.d_1 d_2 ... d_n, 0 \leq d_i < 10$$

biçiminde olup, $0 < q_f < 1$ dir. e ise **üs** adı verilen bir tam sayıdır.

Kesirdeki noktanın pozisyonu, üs değiştirilmek suretiyle sağa veya sola kaydırılabileceğinden ötürü bu formatta ifade edilen sayılara *kayan noktalı sayılar* adı verilmektedir. Örneğin on tabanlı sistemde

$$123.4 = 12.34 \times 10^1 = 1.234 \times 10^2 = 0.1234 \times 10^3$$

kayan nokta sayıları birbirine eşittir ve bir sayının birden fazla kayan nokta gösterimi mevcuttur . Üs değiştirmek suretiyle kesir noktasının sola doğru nasıl kaydırıldığına dikkat edelim.

On tabanlı sistemde bir kayan noktalı sayıyı tanımlayabilmek için *işaret, kesir*(q_f) ve *üs*(e) bilgileri gerekmektedir.

Gözlem 2.1. *Yukarıdaki örnekte görüldüğü üzere aynı sayı birden fazla üs ve mantis ile ifade edilebilir. Diğer deyimle en genel halde kayan nokta gösterimi tek türlü değildir. Ancak normalize edilmiş formatta bu gösterim tek türlüdür: Kesir noktasının sağındaki ilk rakamın sıfırdan farklı olması durumunda kayan nokta formatı, sayının mantis ve üs değerini tek türlü olarak belirler.*

Örneğin

$$123.4 = 0.1234 \times 10^3$$

gösteriminde kesir .1234, taban 10, üs 3 ve sayının inceliği 4 tür.

TANIM 2.5. *$x_f = \pm q_f \times 10^e$ kayan nokta gösteriminde, $q_f = 0.d_1d_2...d_n$ nin d_1 ile gösterilen başlangıç pozisyonundaki rakam sıfırdan farklı ise x_f ye* ***normalize edilmiş sayı*** *adı verilir. Bu durumda $0.1 \leq q_f < 1$ dir. Normalize edilmiş formatta, kayan noktanın sağındaki rakamların sayısına(burada n) x_f nin* ***inceliği****(precision) adı verilir.*

Örneğin 0.1234×10^3 gösterimi 123.4 sayısının *normalize edilmiş kayan nokta* gösterimidir ve gösterimin inceliği dörttür.

2.2.1 $R_f(10)$ sayı sisteminin özellikleri

Kayan nokta sayı sisteminde temsil edilebilecek sayılarının büyüklüklerini ve birbirlerine göre konumlarını yakından görebilmek için bu sayı sistemini yakından incelemek gerekir. Bunun için çok az sayıda nokta içerebilen **özel**

bir kayan nokta sayı sistemi göz önüne alacağız. Özel diyoruz, çünkü hiç bir sistem bu kadar az sayıda nokta içermez. Gerçekçi olmayacak kadar az sayıda nokta içermesine rağmen, örneğimizdeki kayan nokta sistemi ve üzerindeki işlem sonuçları bilgisayarlarda kullanılan gerçek sistemlerin tipik özelliklerini içermektedir.

2.2.2 Özel bir $R_f(10)$ örneği: $\ddot{o}zel\_R_f(10)$

Çok az sayıda eleman içeren ve $\ddot{o}zel\_R_f(10) \subset R$ ile göstereceğimiz on tabanlı bir kayan nokta sayı sistemi gözönüne alalım. Sistemimizin normalize edilmiş

$$x_f = \mp q_f \times 10^e$$

biçiminde sayılardan oluştuğunu ve

$$e = 0, 1, 2$$

ve

$$q_f = 0.d_1d_2, d_1 \neq 0, d_2 = 0, 1, \ldots, 9$$

değerlerini alabileceğini kabul edelim. Teknik bir ifade ile

$$\ddot{O}zel\_\ R_f(10) = \left\{ \begin{array}{c} x_f = \mp q_f \times 10^e |\ q_f = 0.d_1d_2; d_i = 0, 1, \ldots, 9; \\ i = 0, 1; d_1 \neq 0; e = 0, 1, 2 \end{array} \right\}$$

noktalarında oluşmaktadır.

$\ddot{O}zel\_\ R_f(10)$ kayan nokta kümesinin elemanlarını belirleyerek, sayıların birbirlerine göre konumlarını ve $\ddot{o}zel\_\ R_f(10)$ üzerindeki aritmetik işlemlerin özelliklerini incelemeye çalışalım:

- $\ddot{O}zel\_\ R_f(10)$ da temsil edilebilecek en büyük sayı $d_1 = 9, d_2 = 9$ ve $e = 2$ ile pozitif işaretli $x_f = 0.99 \times 10^2 = 99$ sayısıdır. O halde negatif işareti de dikkate aldığımızda

  $$\ddot{o}zel\_R_f(10) \ \subset [-99, 99]$$

  olduğunu görürüz.

- Örnek sistemde her bir üs için temsil edilebilecek pozitif sayılar aşağıdaki tablolarda gösterilmektedir.

$e=0$					
	$d_1=1\rightarrow$	0.10	0.11	$\cdots$	0.19
	$d_1=2\rightarrow$	0.20	0.21	$\cdots$	0.29
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$d_1=9\rightarrow$	0.90	0.91	$\cdots$	0.99

$e=1$					
	$d_1=1\rightarrow$	1.0	1.1	$\cdots$	1.9
	$d_1=2\rightarrow$	2.0	2.1	$\cdots$	2.9
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$d_1=9\rightarrow$	9.0	9.1	$\cdots$	9.9

$e=2$					
	$d_1=1\rightarrow$	10	11	$\cdots$	19
	$d_1=2\rightarrow$	20	21	$\cdots$	29
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$d_1=9\rightarrow$	90	91	$\cdots$	99

  Her bir tabloda 90 adet olmak üzere *özel*$\_R_f(10)$ da 270 adet pozitif sayının yer aldığını görürüz. Negatif sayıların da ilavesiyle örnek kayan nokta sayı sistemimiz toplam 540 adet normalize edilmiş sayıdan oluşmaktadır.

- *Özel*$\_R_f(10)$ da komşu noktalar arasındaki uzaklığın eşit olmadığını ve orjinden uzaklaştıkça komşu noktalar arasındaki mesafenin e nin her bir yeni değeri için 10 katı(taban kadar) arttığını gözlemliyoruz. Ancak e nin aynı değeri için elde edilen ve birbirine komşu olan kayan nokta sayıları arasındaki uzaklığın eşit olduğunu görüyoruz. Örneğin

  - $e=0$ için komşu sayılar arasındaki uzaklık 0.01,
  - $e=1$ için komşu sayılar arasındaki uzaklık 0.1 ve
  - $e=2$ için komşu sayılar arasındaki uzaklık 1 dir ve e nin her bir değeri için elde edilen komşu sayılar arasındaki uzaklık, bir öncekilerin 10(yani taban) katıdır.

- Sıfır sayısının normalize edilmiş bir kayan nokta sayısı olmadığını görüyoruz. $d_1 = 0$ değerine izin verilmesi durumunda sıfır sayısını da sistemde temsil edebiliriz. $d_1 = 0$ değerine karşılık gelen normalize edilmiş formatta ifade edilemeyen sayılara **subnormal** sayılar adı verilmektedir. Bu durumda sistemimizin subnormal sayıları $0.01, 0.02, ..., 0.09$ ve bu sayıların negatifleri ile birlikte $0.0, -0.0$ sayısı olmak üzere toplam 20 adet sayıdır. Burada 0.0 ve -0.0 ın her ikisi de sıfır sayısını temsil etmektedir.

- Temsil edilebilecek en büyük pozitif sayıdan daha büyük bir sayı için **inf**(∞) ve en küçük negatif sayıdan daha küçük bir sayı durumunda $-$**inf**$(-\infty)$ sembolleri uygun subnormal formatta ifade edilirler. Ayrıca reel sayılarda belirsizlik durumları olarak bilinen $0/0$, ∞/∞ için **NaN**, **N**ot **a** **N**umber(sayı değil) gösterimi kullanılır. Örnek sayı sistemimizde

$$99 + 1 = \inf, -99 - 1 = -\inf, 0/0 = \text{NaN}$$

  olarak sembolize edilir. Bu semboller için bellekte özel gösterimler kullanılır.

- *Özel*_ $R_f(10)$ *da kesme ve yuvarlama: Özel*_ $R_f(10)$ sayı sisteminde herhangi iki sayı ile gerçekleştirilen aritmetik işlem sonucu *özel*_ $R_f(10)$ sayı sisteminin bir elemanı değilse(çok büyük ve küçük sayı olma durumu hariç), bu durumda elde edilen sonuç en yakın *özel*_ $R_f(10)$ sayısına **kesme** veya **yuvarlama** prensibine göre yuvarlanır. *Kesme prensibine* göre yuvarlama işlemi aşağıda görüldüğü biçimde noktadan sonraki kısmın kesilmesi suretiyle gerçekleştirilir:

$$10.4 \doteq 10, 10.5 \doteq 10, 10.7 \doteq 10$$

  Yuvarlama prensibinde ise virgülden(veya noktadan) sonraki kısım doğrudan kesilmeyerek, virgülden(veya noktadan) sonraki ilk rakamın tabanın yarısından büyük veya eşit olması veya küçük olması durumlarında farklı yuvarlama işlemleri gerçekleştirilir. Aşağıdaki örnekleri inceleyelim:

$$10.4 \doteq 10, 10.5 \doteq 11, 10.7 \doteq 11$$

Şimdi de yuvarlama işlemlerinin $\ddot{O}zel\_R_f(10)$ üzerindeki aritmetiği nasıl etkilediğini inceleyelim:

- *Özel_$R_f(10)$ da toplama işlemine göre birleşme özelliği geçerli değildir*: *İşlemlerin en yakın sayıya yuvarlama prensibine göre* gerçekleştirildiğini kabul edersek

$$(0.3 + 0.4) + 10 = 0.7 + 10 \doteq 11 \neq 10 \doteq 0.3 + (0.4 + 10)$$

elde ederiz. Çünkü 10.7 sayı sistemimizin bir elemanı değildir ve sistemde bu sayıya en yakın sayı 11 dir. Benzer biçimde $10.4 \doteq 10$ dur.

Eğer *işlemlerin kesme prensibine göre* gerçekleştirildiğini kabul edersek

$$(0.6 + 0.7) + 10 = 11 \neq 10 = 0.6 + (0.7 + 10)$$

elde ederiz, çünkü kesme prensibine göre $11.3 \doteq 11$ ve $10.7 \doteq 10$, $10.6 \doteq 10$ dur.

- *Özel_$R_f(10)$ toplama işlemine göre de kapalı değildir*: Örneğimiz için

$$50, 60 \in \textit{özel}\_R_f(10)$$

fakat

$$110 \notin \textit{özel}\_R_f(10)$$

dur.

- $R -> $ *Özel_$R_f(10)$ dönüşümü ve hata*: $x \in R \backslash R_f(10) \subset [-99, 99]$ olması durumunda, x yerine $x_f \in R_f(10)$ yaklaşımı kullanılır. Söz konusu yaklaşım kesme veya en yakın sayıya yuvarlama esasına göre gerçekleştirilir: $x = 11.6$ sayısı gözönüne aldığımız örnek kayan nokta sayı sisteminde temsil edilmemektedir. Bu durumda kesme prensibine göre $x_f = 11$ yaklaşımı kullanılır. Bu durumda oluşan mutlak hata

$$\Delta x = 11.6 - 11 = 0.6$$

ve bağıl hata ise

$$\epsilon_b(x) = \frac{\Delta x}{x} = \frac{0.6}{11.6} \doteq 0.0517$$

dir. En yakın sayıya yuvarlama prensibine göre ise kesir kısım atılarak, bir öndeki rakama 1 ilave etmek suretiyle $x_f = 12$ alınır. Bu durumda oluşan mutlak hata

$$\Delta x = 11.6 - 12 = -0.4$$

ve bağıl hata ise

$$\epsilon_b(x) = \frac{\Delta x}{x} = \frac{-0.4}{11.6} \doteq -0.0345$$

dir.

Şimdi de gerçek bir kayan nokta sistemine dönüşümde oluşabilen hataları inceleyelim:

2.2.3 $R- > R_f(10)$ dönüşümü ve hata

$x \in R$ yerine $x_f \in R_f$ yaklaşımının alınması durumunda yuvarlama hataları oluşur:

- On tabanlı sistemde

  $$x = (0.d_1 \cdots d_n d_{n+1} \cdots) \times 10^e, d_1 \neq 0, 0 \leq d_i \leq 9, i = 1, 2, ..., 9$$

  sayısı için *kesme yöntemine göre* R_f deki yaklaşım

  $$x_f = (0.d_1 \cdots d_n) \times 10^e$$

  olarak tanımlanır. Bu durumda oluşan mutlak hatanın üst sınırı

  $$\begin{aligned} \Delta x &= x - x_f \\ &= (0.0...0 d_{n+1} \cdots) \times 10^e \\ &= (d_{n+1}.d_{n+2} \cdots) \times 10^{-(n+1)} \times 10^e \\ &\leq 10 \times 10^{-n-1} \times 10^e = 10^{e-n} \end{aligned}$$

  olarak tahmin edilebilir(burada $d_{n+1}.d_{n+2} \cdots \leq 10$ eşitsizliğini kullandık).

- Yine on tabanlı sistemde

  $$x = (0.d_1 \cdots d_n d_{n+1} \cdots) \times 10^e, 0 \leq d_i \leq 9, d_1 \neq 0, i = 1, 2, ..., 9$$

  sayısı için *en yakın sayıya yuvarlama esasına göre* R_f de

  $$x_f = \begin{cases} (0.d_1 \cdots d_n) \times 10^e, & d_{n+1} \leq 4 \\ (0.d_1 \cdots d_n + 10^{-n}) \times 10^e & d_{n+1} \geq 5 \end{cases}$$

olarak tanımlanır. Örneğin $d_{n+1} \leq 4$ olması durumunda

$$\begin{aligned}\Delta x &= x - x_f \\ &= (0.0_1 \cdots 0_n d_{n+1} \cdots) \times 10^e \\ &\leq (d_{n+1}.d_{n+2} \cdots) \times 10^{-(n+1)+e} \\ &\leq 5 \times 10^{-(n+1)+e} = \frac{1}{2} \times 10^{e-n}\end{aligned}$$

olarak tahmin edilebilir.$d_{n+1} \geq 5$ olması durumunda ise

$$\begin{aligned}\Delta x &= x - x_f \\ &= (0.d_1 \cdots d_n d_{n+1} \cdots) \times 10^e - (0.d_1 \cdots d_n + 10^{-n}) \times 10^e \\ &= (0.0_1 \cdots 0_n d_{n+1} \cdots) \times 10^e - 1 \times 10^{e-n} \\ &= (0.d_{n+1} \cdots) \times 10^{e-n} - 10^{e-n} \\ &= -[1 - (0.d_{n+1} \cdots)] \times 10^{e-n}\end{aligned}$$

olup,

$$|\Delta x| = [1 - (0.d_{n+1} \cdots)] \times 10^{e-n} \leq \frac{1}{2} \times 10^{e-n} \tag{2.3}$$

elde ederiz. O halde en yakın sayıya göre yuvarlama esasına göre d_{n+1} değerinden bağımsız olarak

$$|\Delta x| \leq \frac{1}{2} \times 10^{e-n}$$

elde ederiz.

- Öte yandan yukarıda incelenen yuvarlama işlemin gerçekleştirilme biçiminden bağımsız olarak

$$\begin{aligned}|\varepsilon_b(x)| &= \frac{|\Delta x|}{|x|} \\ &\leq \frac{\frac{1}{2} \times 10^{e-n}}{(0.d_1 \cdots d_n d_{n+1} \cdots) \times 10^e} \\ &\leq \frac{1}{2} \times 10^{-n+1} = 5 \times 10^{-n}\end{aligned} \tag{2.4}$$

elde ederiz çünkü $d_1 \geq 1$ olduğundan

$$0.d_1 \cdots d_n d_{n+1} \cdots >= 10^{-1}$$

dir. Bağıl hata için elde ettiğimiz üst sınırı $\epsilon_{\max}$ ile göstereceğiz. O halde 10 tabanlı sayı sistemi için

$$\epsilon_{\max} = 5 \times 10^{-n}$$

dir.

- 2 tabalı sistem için

$$\epsilon_{\max} = \frac{1}{2} \times 2^{-n+1} = 2^{-n}$$

olarak elde edililir ve bu değer *bilgisayar* epsilonu olarak adlandırılır: $|\varepsilon_b(x)| \leq \epsilon_{\max}$ dır.

ÖRNEK 2.1.

$$\begin{aligned} x &= 3.14159265358979 \\ &= 0.314159265358979 \times 10^1 \end{aligned}$$

için en yakın sayıya yuvarlama prensibine göre elde edilen

$$x_f \doteq 0.31416 \times 10^1$$

yaklaşımı ile oluşan mutlak ve bağıl hatayı belirleyiniz ve (2.3) ,(2.4) ile verilen üst sınırların geçerli olduğunu gözlemleyiniz.

$$\begin{aligned} \Delta x &= x - x_f \\ &= 7.34641020683213 \times 10^{-6} \\ &= 0.0734641020683213 \times 10^{-4} \\ &\leq \frac{1}{2} \times 10^{1-5} = \frac{1}{2} \times 10^{e-n} \end{aligned}$$

ve

$$\begin{aligned} \varepsilon_b(x) &= \frac{\Delta x}{x} \\ &\leq \frac{\frac{1}{2} \times 10^{1-5}}{0.314159265358979 \times 10^1} \\ &< \frac{1}{2} \times 10^{-4} \end{aligned}$$

olup, (2.3) ,(2.4) ile verilen üst sınırlar geçerlidir.

Bir diğer hata kaynağı on tabanlı sistemden iki tabanlı sisteme dönüşümde oluşan hatalardır.

2.3 $R_f(10) \longleftrightarrow R_f(2)$ Taban dönüşümleri ve ilgili hatalar

Bilgisayarların çoğu iki tabanlı sayı sistemini kullanır. Kullandığımız on tabanlı sayı sisteminde sonlu sayıda rakamla temsil edilebilen bir sayının bilgisayarların kullandıkları iki tabanlı sayı sisteminde sonlu sayıda rakamla ifade edilememe ihtimali söz konusudur.

Bu noktayı açıklamak için öncelikle göz önüne aldığımız bilgisayar sisteminin iki tabanlı sayı sistemini kullandığını kabul ederek, iki tabanlı ve on tabanlı sayı sistemleri arasındaki dönüşümün nasıl gerçekleştirildiğini inceleyelim:

2.3.1 $R_f(2)$ ve $R_f(2) -> R_f(10)$ dönüşümü

İkili sistemde *sonlu sayıda* basamağa sahip herhangi bir

$$x = (c_1 c_2 \cdots c_{n+1}.d_1 d_2 \cdots d_m)_2$$

reel sayısı, uygun $m \geq 0, n \geq 0$ tamsayıları için (2.1) e benzer olarak

$$V = \{2^n, 2^{n-1}, \cdots, 2^0, 2^{-1}, \cdots, 2^{-m}\} \tag{2.5}$$

kümesinin veya taban elemanlarını lineer kombinasyonu olarak ifade edilebilir.

Diyer deyimle uygun $c_i, d_i \in \{0, 1\}$ ve $n \geq 0, m \geq 0$ tamsayıları için

$$\begin{aligned} x &= (c_1 c_2 \cdots c_{n+1}.d_1 d_2 \cdots d_m)_2 \\ &= c_1 2^n + c_2 2^{n-1} + \cdots + c_{n+1} 2^0 \\ &\quad + d_1 2^{-1} + \cdots + d_{m-1} 2^{-m+1} + d_m 2^{-m} \end{aligned} \tag{2.6}$$

biçiminde V kümesinin elemanlarının lineer bileşimi olarak yazılabilir.

$R_f(2)$ ile gösterdiğimiz iki tabanlı sistemde kullanılan rakamlar 0 ve 1 olup, taban ise 2 dir.

İkili sayı sisteminde herhangi bir tam sayı $c_i \in \{0, 1\}$ rakamları ve 2 nin azalan kuvvetleri cinsinden ifade edilerek on tabanlı sisteme dönüştürülür. Örneğin

$$\begin{aligned} (101101)_2 &= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 32 + 8 + 4 + 1 \\ &= 4 \times 10^1 + 5 \times 10^0 \\ &= (45)_{10} \end{aligned}$$

İki tabanlı sistemdeki kesirli bir sayı da benzer biçimde on tabanlı sisteme dönüştürülebilir. Örneğin

$$\begin{aligned}(0.101111)_2 \times 2^3 &= (101.111)_2 \\ &= 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3} \\ &= 4 + 1 + 1/2 + 1/4 + 1/8 = 5.875.\end{aligned}$$

olarak elde edilir.

Genelde

$$P(x) = c_1 x^n + c_2 x^{n-1} + \cdots + c_n x + c_{n+1}$$

ve

$$Q(x) = d_m x^m + d_{m-1} x^{m-1} + \cdots + d_1$$

olmak üzere

$$(c_1 c_2 \cdots c_{n+1}.d_1 d_2 \cdots d_m)_2$$

iki tabanlı gösteriminin on tabanlı karşılığını (2.6) den

$$P(2) + \frac{1}{2} Q(1/2) \tag{2.7}$$

olarak ifade edebiliriz. $P(x)$ polinomunun $x = 2$ noktasında ve $Q(x)$ polinomunun $x = 1/2$ noktasındaki değerini ise MATLAB/Octave **Polyval** fonksiyonu ile hesaplıyoruz. Aynı işlemi 3. Bölümde inceleyeceğimiz Horner yöntemi ile de gerçekleştirebiliriz.

$$c = [c_1, c_2, \cdots c_{n+1}], d = [d_1, d_2, \cdots, d_m]$$

olmak üzere

$$(c.d)_2$$

ikili sayısının on tabanlı gösterimini hesaplayan Program 2.1 aşağıda verilmektedir. İkili sistemdeki sayının kesir kısmı mevcut değilse $d = 0$, tam kısmı mevcut değilse $c = 0$ olarak girilmelidir.

Örnek Uygulamalar:

- $(101101)_2$ sayısının on tabanlı gösterimini Program 2.1 ile belirleyiniz.

$>> c = [1\ 0\ 1\ 1\ 0\ 1];$
$>> d = 0;$
$>>$ ikidenon(c, d)
Onlu taban gosterim: 45

```
%-------------------------------------------------------
function ikidenon(ikitam,ikikesir)
 % ikili sistemden onlu sisteme donusum
 % ikitam, tam kesime ait 0 ve 1 ler dizisi
 % ikikesir: kesirli kesime ait 0 ve 1 ler dizisi, ec.

x0=2;
tamon=polyval(ikitam,x0); %
x0=1/2;
ikikesir=fliplr(ikikesir); % kesir  indislerinin tersten dizilimi
tamkesir=polyval(ikikesir,x0);
sonuc=tamon+1/2*tamkesir;
fprintf('Onlu taban gosterim: '); disp(num2str(sonuc));

%-------------------------------------------------------
```

Program 2.1: Ikili sistemden onlu sisteme dönüşüm uygulaması

- $(101.111)_2$ sayısının on tabanlı gösterimini Program 2.1 ile belirleyiniz.

  $>> c = [1\ 0\ 1]; d = [1\ 1\ 1];$
  $>>$ ikidenon(c, d)
  Onlu taban gosterim:5.875

- $(0.001)_2$ sayısının on tabanlı gösterimini Program 2.1 ile belirleyiniz.

  $>> c = 0; d = [0\ 0\ 1];$

  $>>$ ikidenon(c, d)

  Onlu taban gosterim: 0.125

2.3.2 $R_f(10) -> R_f(2)$ dönüşümü

- *Tamsayı dönüşümü:* Bu dönüşüm için aşağıdaki örneği inceleyelim.

ÖRNEK 2.2. 1964 *sayısını iki tabanlı gösterimini elde ediniz.*

Çözüm.

$$2^{10} = 1024$$

olduğu dikkate alındığında

$$1964 = c_1 2^{10} + c_2 2^9 + \cdots c_{10} 2^1 + c_{11} 2^0$$

sağlanacak biçimdeki

$$c_i \in \{0, 1\}, i = 0, 1, ..., 10$$

sabitlerini belirlemeliyiz. En son sabit olan c_{11}'in 1964'ün ikiye bölümünde elde edilen kalan olduğu açıktır. Yani

$$1964 = 2 \times 982 + c_{11}$$

dur ve 1964 çift sayı olduğu için $c_{11} = 0$ dır. Şimdi ise eşitliğin her iki tarafını ikiye bölerek elde edilen

$$982 = c_1 2^9 + c_2 2^8 + \cdots + c_9 2^1 + c_{10}$$

ifadesinden de c_{10}'nun 982'nin 2'ye bölümünden elde edilen kalan olduğuna dikkat edelim. Yani

$$982 = 2 \times 491 + c_{10}$$

dur. Benzer işlemler tekrar edilerek

$$\begin{aligned}
1964 &= 2 \times 982 + c_{11} \Rightarrow c_{11} = 0 \\
982 &= 2 \times 491 + c_{10} \Rightarrow c_{10} = 0 \\
491 &= 2 \times 245 + c_9 \Rightarrow c_9 = 1 \\
245 &= 2 \times 122 + c_8 \Rightarrow c_8 = 1 \\
122 &= 2 \times 61 + c_7 \Rightarrow c_7 = 0 \\
61 &= 2 \times 30 + c_6 \Rightarrow c_6 = 1 \\
30 &= 2 \times 15 + c_5 \Rightarrow c_5 = 0 \\
15 &= 2 \times 7 + c_4 \Rightarrow c_4 = 1 \\
7 &= 2 \times 3 + c_3 \Rightarrow c_3 = 1 \\
3 &= 2 \times 1 + c_2 \Rightarrow c_2 = 1 \\
1 &= 2 \times 0 + c_1 \Rightarrow c_1 = 1
\end{aligned}$$

elde edilir. O halde

$$\begin{aligned} 1964 &= 1\times 10^3 + 9\times 10^2 + 6\times 10^1 + 4\times 10^0 \\ &= c_1 2^{10} + c_2 2^9 + \cdots + c_9 2^2 + c_{10} 2^1 + c_{11} \\ &= 1\times 2^{10} + 1\times 2^9 + 1\times 2^8 + 1\times 2^7 \\ &+ 0\times 2^6 + 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 \\ &+ 0\times 2^1 + 0\times 2^0 \end{aligned}$$

olup,

$$1964 = (11110101100)_2$$

iki tabanlı gösterimini elde ederiz.

Özetle on tabanlı sayıyı iki tabanlı bir sayıya dönüştürmek için uygulamamız gereken pratik kural şudur:

Sayı ikiye bölünerek kalanı not ettikten sonra bölüm kısmı ile aynı işlemi tekrarlanır. Elde edilen kalanların sondan başa doğru yan yana dizilimi ilgili sayının iki tabanlı gösterimidir.

Program 2.2 yukarıda açıklanan yöntemi kullanmak suretiyle, girilen on tabanlı pozitif bir sayının iki tabanlı gösterimini belirler:

>> tamdaniki(1964)

ans =

1 1 1 1 0 1 0 1 1 0 0

Program 2.2 ile Tablo 2.1 ile verilen dönüşümleri elde ederiz:

On tabanlı sayı	İki tabanlı gösterimi
2	10
3	11
4	100
5	101
10	1010
100	1100100
1000	1111101000

Tablo 2.1: On tabanlı bazı tam sayıların ikili sistem gösterimleri

- *Sıfır ve bir arasındaki on tabanlı kesirli bir sayının iki tabanına dönüşümü:* On tabanlı sistemde ifade edilen sıfır ve bir arasındaki herhangi bir kesirli sayı 10 nun negatif kuvvetleri yardımıyla ifade edilebileceği gibi,

```
%-------------------------------------------------------
function ikili=tamdaniki(sayi)
% Onlu sistem tam sayını ikili sisteme dönüştürür, ec.

 sayac=1;
  while sayi>=2
    kalan=mod(sayi,2);
    ikili(sayac)=kalan; % kalanlar   dizisi
    sayi=(sayi-kalan)/2;
    sayac=sayac+1;
  end
  ikili(sayac)=sayi; % son kalan eleman
 ikili=ters_cevir(ikili);
function ters_ikili=ters_cevir(ikili)
  m=length(ikili);
  for i=1:m                      % ikili dizisinin tersten dizilimi
    ters_ikili(i)=ikili(m+1-i);  % MATLAB/Octave   fliplr fonksiyonu
  end                            % aynı  işlemi gerçekleştirir
%-------------------------------------------------------
```

Program 2.2: Onlu sistem tam sayısını ikili sisteme dönüştürme uygulaması

iki tabanlı sayı sisteminde de 2 nin negatif kuvvetleri yardımıyla ifade edilebilir: Kesirli sayı dönüşümünde $[x]$ notasyonu ile x sayısının tam kısmını ve $\{x\}$ ile de kesirli kısmını gösterelim. Kesirli sayı taban dönüşümünü aşağıdaki örnek üzerinde inceleyelim:

ÖRNEK 2.3. $A = 0.125$ *sayısının ikili sistemdeki gösterimini belirleyiniz.*

Çözüm.

On tabanlı sistemde

$$A = 0.125 = 1 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

olarak ifade edilir. Aynı sayıyı iki tabanlı sistemde

$$\begin{aligned} A &= 0.125 = (0.d_1d_2d_3d_4\ldots) \\ &= d_1 \times 2^{-1} + d_2 \times 2^{-2} + d_3 \times 2^{-3} + d_4 \times 2^{-4} + \cdots \end{aligned}$$

şeklinde ifade ettiğimizi varsayalım. İfadenin her iki yanını 2 ile çarparsak

$$2A = 0.250 = d_1 + d_2 \times 2^{-1} + d_3 \times 2^{-2} + d_4 \times 2^{-3} + \cdots$$

elde ederiz. Burada

$$d_1 = [2A] = 0$$

ve

$$k_1 = \{2A\} = 0.250 = d_2 \times 2^{-1} + d_3 \times 2^{-2} + d_4 \times 2^{-3} + \cdots$$

dir. Her iki yanı tekrar 2 ile çarparak

$$2k_1 = 0.50 = d_2 + d_3 \times 2^{-1} + d_4 \times 2^{-2} + \cdots$$

$$d_2 = [2k_1] = 0$$

dır. Benzer biçimde

$$k_2 = \{2k_1\} = 0.50 = d_3 \times 2^{-1} + d_4 \times 2^{-2} + \cdots$$

$$2k_2 = 1.0 = d_3 + d_4 \times 2^{-1} + \cdots$$

$$d_3 = [2k_2] = 1$$

ve

$$k_3 = \{2k_2\} = 0$$

elde edilir. Böylece kesir kısmı sıfır olana kadar işleme devam ettirilerek elde edilen $d_i, i = 1, 2, \ldots$değerleri kaydedilir. O halde

$$A = 0.125 = (0.001)_2$$

iki tabanlı gösterimi elde edilir.

Yukarıdaki örnekte özetlenen sıfır ve bir arasındaki kesirli bir sayının iki tabanına dönüşümünü gerçekleştiren Program 2.3 aşağıda verilmektedir.

Program 2.3 ikili sistem gösteriminin noktadan(virgülden) sonraki basamaklarını vermektedir:

>> kesirdeniki(0.125)

ans = 001

O halde istenilen gösterim 0.001 dir.

Program 2.3 ile Tablo 2.2 ile verilen dönüşümleri elde ederiz:

Tablo 2.2 de 0.1 sayısına karşılık gelen ikili gösterimin, 0011 rakamlar grubunun tekrar etmesiyle, devirli bir sayı olduğu görülmektedir.

- *On tabanlı herhangi bir kesirli bir sayının iki tabanına dönüşümü:*

ÖRNEK 2.4. 1964.125 *sayısının iki tabanlı gösterimini elde ediniz.*

```
%-------------------------------------------------------
  function d=kesirdeniki(sayi)
 % On tabanlı sayısını 0<sayi<1, iki tabanlı sisteme dönüştürür, ec.

  tamsayi=fix(sayi);kesirsayi=sayi-tamsayi;
  kesir(1)=2*kesirsayi-fix(2*kesirsayi);
  d(1)=fix(2*kesirsayi);
  i=1;
  while (kesir(i)>0)&&(i<23)
   i=i+1;
   d(i)=fix(2*kesir(i-1)) ;
   kesir(i)=2*kesir(i-1)-fix(2*kesir(i-1));
  end

%-------------------------------------------------------
```

Program 2.3: On tabanlıkesirli sayısının ikili sisteme dönüşüm uygulaması

Kesirli sayı	İki tabanlı gösterimi
0.125	0.001
0.250	0.01
0.5	0.1
0.75	0.11
0.1	$0.0\ \overline{0011}$

Tablo 2.2: Bazı kesirli sayıların ikili sistem gösterimleri.

Çözüm.

Bunun için yapmamız gereken işlem, Örnek 2.2 ve Örnek 2.3'e ait sonuçları birleştirmektir. On tabanlı sistemde

$$\begin{aligned} 1964.125 &= 1\times 10^3+9\times 10^2+6\times 10^1+4\times 10^0 \\ &\quad +1\times 10^{-1}+2\times 10^{-2}+5\times 10^{-3} \end{aligned}$$

olacak şekilde ifade edildiği gibi, tam ve kesirli kısımlara karşılık gelen iki tabanlı gösterimler de birleştirildiğinde

$$\begin{aligned} 1964.125 &= (1111010100.001)_2 \\ &= (1.111010100001)_2\times 2^9 \end{aligned}$$

normalize edilmiş kayan nokta gösterimi elde edilir.

On tabanlı herhangi bir sayıyı iki tabanlı sisteme dönüştüren Program 2.4 aşağıda verilmektedir.

```
%-----------------------------------------------------------
function ondaniki(sayi)
% On tabanlı herhangi bir sayıyı ikili tabana dönüştürür ve
% sonucu kontrol eder, ec.

tamsayi=fix(sayi);kesirsayi=sayi-tamsayi;
 if tamsayi>0
   k=tamdaniki(tamsayi);
 else
  k=0;
 end
 if kesirsayi>0
  d=kesirdeniki(kesirsayi);
 else
  d=0;
 end
fprintf('ikili taban gosterimi:  ');
sayi=strcat(num2str(k),'.');
sayi=strcat(sayi,num2str(d));
display(num2str(sayi));
fprintf('Kontrol : ');
ikidenon(k,d);

%-----------------------------------------------------------
```

Program 2.4: On tabanlı sistemden iki tabanli sisteme dönüşüm uygulaması

Program 2.4 ile elde edilen iki tabanlı gösterimin doğruluğu da kontrol edilmektedir. Bu amaçla onlu sistemdeki sayının tam ve kesirli kısımlarının ikili sistem karşılıkları Program 2.1 fonksiyon programına gönderilerek, on tabanlı gösterim tekrar elde edilmektedir.

>> ondaniki(1964.125)
ikili taban gosterimi: 11110101100.001
Kontrol:
Onlu taban gosterimi: 1964.125
>> ondaniki(21.75)
ikili taban gosterimi: 10101.11

Kontrol:

Onlu taban gosterimi: 21.75

Böylece on tabanlı sistemdeki kesirli bir sayının ikili sistemdeki karşılığını nasıl elde edeceğimizi öğrenmiş bulunuyoruz. Şimdi de $R_f(2)$ sayı sisteminin elemanlarının bilgisayar belleğinde nasıl temsil edildiklerini inceleyelim.

2.4 $R_f(2)$ nin bellek gösterimi

$IEEE754$ standartı adı verilen uluslararası standarta göre x_f ile gösterilen bir kayan nokta sayısı, 32 bit formatta

1 bit(sayı işareti)	8 bit(Üs)	23 bit (Mantis)

ve 64 bit formatta ise

1 bit(sayı işareti)	11 bit(Üs)	52 bit (Mantis)

bellek alanlarında temsil edilir. MATLAB/Octave 64 bit formatı kullanır[8].

İşaret bitindeki '0', ilgili sayının pozitif ve '1' ise negatif olması anlamına gelir.

İkili sistemde 32 bit formatında temsil edilebilecek normalize edilmiş

$$x_f = \pm(1.d_1d_2\ldots d_{23}) \times 2^e$$

sayısını göz önüne alalım. Kapalı bit gösterimi(implicit bit representation) adı verilen bir standarta göre başlangıç rakamı olan 1 sisteme kaydedilmez.

Orjine göre simetrik olmayan(biased) formatta negatif üsleri temsil etmek için işaret biti kullanılmaz ve

$$\text{depolanan üs}(e_d)=\text{gerçek üs}(e) + 127$$

kuralına göre üs değerleri depolanır. Burada 127 ise öteleme sabiti olarak adlandırılır. 32 bit formatına göre $-126 \leq e \leq 127$ dir. Bu kurala göre aşağıdaki tabloda belirtilen gerçek üsler, karşılarındaki belirtilen depolanan

üs değerleri ile temsil edilirler:

gerçek üs(e)	depolanan üs($e_d(10)$)→	ikili sistem karşılığı($e_d(2)$)
$-126 \rightarrow$	1	00000001
$-125 \rightarrow$	2	00000010
$\vdots$	$\vdots$	$\vdots$
$1 \rightarrow$	128	10000000
$127 \rightarrow$	254	11111110

ÖRNEK 2.5.

$$1.964125 \times 10^3 = (1.111010100001)_2 \times 2^9$$

sayısının 32 *bit formatta bellek alanına yerleşimini belirleyiniz.*

Çözüm.

Yukarıdaki tabloya göre verilen sayının gerçek üssü olan $e = 9$ değeri $e_d = 136$ depolanan üs değerine karşılık gelmektedir. Öte yandan

$$136 = (10001000)_2$$

dir. Mantisin başlangıç bitindeki 1 değeri kapalı bit gösterimine göre depolanmaz. Kesir noktası da bellek gösteriminde yer almaz. Sadece kesir kısmı olan

$$111010100001$$

değeri 23 bitlik alana sağa yaslı olarak depolanır. Sayı pozitif olduğu için işaret biti 0 olmalıdır. Dolayısıyla

0	1	0	0	0	1	0	0	0	**0**	1	1	1	0	1	0	1	0	0	0	0	1

gösterimini elde ederiz, burada **0** =00000000000 dır.

2.4.1 $R_f(2) - 32$-bit sisteminde en büyük ve en küçük sayılar ve MATLAB/Octave gösterimleri

Bu formatta 8 bitlik üs alanında ikili sisteme göre temsil edilebilecek en büyük üs

$$\begin{aligned} e_d(2) &= (11111110)_2 \\ &= 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 \\ &\quad +1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 + 1 \times 2^7 \\ &= 254 \end{aligned}$$

tür ve yukarıdaki tabloya göre depolanan bu değer $e = 127$ gerçek üs değerini temsil eder.

Ayrıca en büyük mantis değeri

$$\begin{aligned}(1.1_1 1_2 \dots 1_{23})_2 &= 1 + 1/2^1 + 1/2^2 + \cdots + 1/2^{23} \\ &= \frac{1-(1/2)^{24}}{1-1/2} = (2 - 2^{-23})\end{aligned}$$

olup, en büyük pozitif kayan noktalı sayı

$$\begin{aligned}x_f &= (2 - 2^{-23}) \times 2^{127} = 3.402823466385289e + 038 \\ &= \boxed{0 \mid 11111110 \mid 111...1}\end{aligned}$$

dir.

Bu sayı MATLAB/Octave ortamında

>> realmax('single')

ans =

3.4028235e + 038

olarak elde edilir.

Benzer olarak en küçük pozitif değer ise üssün mutlak değerce alabileceği en büyük negatif sayıya(-126) karşılık gelir.-126 için depolanan değer, yukarıdaki tablodan 1 e eşittir. O halde ikili sistemde

$$e_d = (00000001)_2$$

olup,

$$x_f = 2^{-126} = 1.175494350822288e - 038$$

sayısının 32-bit formatta bellek gösterimi

0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	**0**

olarak elde edilir, burada $\mathbf{0} = (0_1 0_2 ... 0_{23})$ tür.

Bu sayı MATLAB/Octave ortamında

>> realmin('single')

ans =

1.1754944e − 038

olarak elde edilir.

Benzer biçimde

0	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	**0**

bellek gösterimi ∞ sembolünü ve

1	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	**0**

ise $-\infty$ sembolünü temsil eder. $e = (11111111)_2$ ve mantis kısmında herhangi bir bit değerinin sıfırdan farklı olacak biçimde depolanan değer ise **NaN** sembolünü temsil eder.

2.4.2 $R_f(2) - 64$-bit sisteminde en büyük ve en küçük sayılar ve MATLAB/Octave gösterimleri

İkili sistemde 64 bit formatında temsil edilebilecek normalize edilmiş

$$x_f = \pm(1.d_1d_2\ldots d_{52}) \times 2^e$$

sayısını göz önüne alalım

64−bit formatında da orjine göre simetrik olmayan(biased) formatta negatif üsleri temsil etmek için işaret biti kullanılmaz, ancak bu durumda

$$\text{depolanan üs}(e_d)=\text{gerçek üs}(e) + \textit{1023}$$

kuralına göre üs değerleri depolanır. Burada 1023 ise öteleme sabiti olarak adlandırılır. 64 bit formatına göre $-1022 \leq e \leq 1023$ dür. Bu kurala göre aşağıdaki tabloda belirtilen depolanan üsler, karşılarındaki gerçek üs değerlerini temsil ederler:

gerçek üs(e)	depolanan üs(e_d)	ikili sistem karşılığı
$-1022 \rightarrow$	1	00000000001
$-1021 \rightarrow$	2	00000000010
$\vdots$	$\vdots$	
$0 \rightarrow$	1023	01111111111
$1 \rightarrow$	1024	10000000000
$\vdots$	$\vdots$	$\vdots$
$1023 \rightarrow$	2046	11111111110

Pozitif bir sayı için $64-$bit formatta 11 bitlik üs alanında ikili sisteme göre temsil edilebilecek en büyük üs

$$e = (11111111110)_2 = (2046)_{10} \rightarrow 1023$$

ve en küçük üs ise

$$e = (00000000001)_2 = (1)_{10} \rightarrow -1022$$

dir. Ayrıca en büyük mantis değeri, 2 den bir önceki sayı olan

$$\begin{aligned}(1.1_1 1_2 \cdots 1_{52})_2 &= 1 + 1/2^1 + 1/2^2 + \cdots + 1/2^{52} \\ &= \frac{1-(1/2)^{53}}{1-1/2} = (2 - 2^{-52})\end{aligned}$$

değeridir ve en büyük pozitif kayan noktalı sayı ise

$$x_f = (2 - 2^{-52}) \times 2^{1023} = 1.797693134862316e + 308$$

olup bu sayının bellek gösterimi

0	$1_1 1_2 ... 1_{10} 0_{11}$	$1_1 1_2 1_3 ... 1_{52}$

elde edilir.

Bu sayı MATLAB/Octave ortamında yukarıda belirtildiği üzere

>> realmax
ans =
1.797693134862316e + 308
olarak elde edilir.

Benzer olarak en küçük pozitif değer ise üssün alabileceği mutlak değerce en büyük negatif sayıya(-1022) karşılık gelir ve

$$x_f = 2^{-1022} = 2.225073858507201e - 308$$

olup, ikili sistem karşılığı

0	0000000001	**0**

dır, burada $\mathbf{0} = (0_1 0_2 ... 0_{52})$ dir. Bu sayı yukarıda da belirtildiği gibi MATLAB/Octave ortamında

$>>$ realmin
ans $=$
2.225073858507201e $-$ 308
olarak elde edilir.
$\mp\infty$ ve NaN gösterimleri 32$-$bit formatta olduğu gibidir.

2.4.3 MATLAB/Octave üzerinde $R_f(2) - 64$ bit aritmetiği ve oluşabilen hatalar

Bilgisayarların kullandığı kayan nokta sayı sistemi de yukarıda örnekte özelliklerini incelediğimiz Özel sistemin özelliklerini taşır. Bu özellikleri bu dökümanı hazırladığımız bilgisayarın kullandığı kayan nokta sayı sistemi üzerinde ve MATLAB/Octave üzerinde kısaca inceleyelim:

- $R_f(2) - 64$ *-bit sonlu sayıda elemandan oluşur:*

  $R_f(2) - 64$ sisteminde, kullandığımız bilgisayar

  $$[-1.7977E + 308, 1.7977E + 308]$$

  aralığı içerisindeki kayan nokta sayılarından oluşan sonlu bir sayı kümesi ile çalışır. Herhangi bir işlem sonucunun bu aralıkta yer alan sayıdan daha büyük bir sayı olması durumunda sonuç *sonsuz=inf* olarak yorumlanır. Örneğin MATLAB/Octave ortamında

  $>>1.797693134862316E + 308 + 1E292$

  $$ans = \text{inf}$$

  sonsuz(inf-infinity) olarak yorumlanmaktadır. Bilgisayarımızda yer

  alan ve sıfırdan büyük normalize edilmiş ilk pozitif sayıyı ise *realmin* komutuyla belirlenebilir:

  $-$ $>>$ x=realmin

  $x = 2.2251E - 308$.

– Sıfır ile kayan nokta sayı sisteminde yer alan bu en küçük sayı arasında oluşan bir işlem sonucu bazı programlarda çok küçük sayı (*underflow*) hatası olarak algılanır ve genelde sonuç sıfır olarak kabul edilerek işleme devam edilir. Bazı uygulamalar genişletilmiş kayan nokta sistemi adı verilen genişletilmiş sistemi kullanarak $[0, realmin]$ aralığında sonlu sayıda *denormal* veya *subnormal* adı verilen sayının temsiline de izin veriler[7]. Bu işlemi $q_f = d_0.d_1d_2...d_n$ gösteriminde d_0'ın da sıfıra eşit olmasına izin vermek suretiyle gerçekleştirir. Örneğin MATLAB/Octave ortamında *realmin* ile temsil edilebilen sayıdan daha küçük subnormal sayılara izin verilmektedir:

$>> 1e-315$

ans =

1.0000e − 315

Fakat

$>> 1e-324$

ans =

0

dır. Bunlara ilaveten belirsizlik adını verdiğimiz tanımlı olmayan (0/0, ∞/∞ gibi) bir işlem sonucu ise *NaN*(Not a Number- sayı değil) olarak kabul edilir.

$>> 0/0$

ans =

NaN

- *Hiç bir irrasyonel sayı veya devirli rasyonel sayı* R_f (10) *da yer almaz(Her bir eleman için sınırlı bellek alanı):*

  $\pi, e, \sqrt{2}$ gibi hiç bir reel sayı sonlu basamaklı q_f ile $x_f = \pm q_f \times 10^E$ biçiminde ifade edilemez. Bu durumda irrasyonel sayılarla işlem yapmak yerine, bu sayılara en yakın olan ve R_f (10) da temsil edilebilen sayılarla çalışılır ve sonuç olarak yuvarlama hatası oluşur. Öte yandan $1/3 = 0.\overline{3}$ devirli sayısı da R_f (10) da yer almaz çünkü sonlu basamaklı bir mantise sahip değildir.

MATLAB/Octave ortamında

$>>$ sin(pi)

ans $=$

$1.2246e-016$ elde edilir, sıfıra eşit olması beklenen bu sonucun sebebi $R_f\ (10)$ daki pi nin tam olarak R deki π ye eşit olmamasıdır.

- $R_f(2) - 64$ *te toplama işleminin birleşme özelliği yoktur:*

  Örneğin MATLAB/Octave ortamında

  $>> 1-(1/3+1/3+1/3)$

  ans $=$

  0

  Fakat

  $>> (1-1/3-1/3)-1/3$

  ans $=$

  $1.1102e-016$ olarak elde edilir.

- *Kayan nokta sayı sisteminde sayılar sıfır noktasının komşuluğunda daha sıkça serpiştirilmiştir ve sonuç olarak işlemlerin gerçekleştirilme sırasına göre farklı yuvarlama hataları oluşabilir ve bu durum işlem sonucunu değiştirebilir:*

  Bu sonucu MATLAB/Octave yazılımlarında kullanılan $eps()$ fonksiyonu yardımıyla gözlemleyebiliriz. Bu fonksiyon $eps(x)$ biçimindeki kullanımıyla, x sayısı ile bu sayıya en yakın bilgisayar sistemindeki sayı arasındaki uzaklığı verir. Tablo 2.3 de bu sonucu gözlemlemeye çalışıyoruz. Tablonun sol sütununda farklı büyüklükte sayılar yer almakta ve sağ sütununda ise bu sayılar ile bunlara en yakın bilgisayar sayı sistemindeki sayılar arasındaki uzaklık listelenmektedir:

  Tablo 2.3 den x değerleri büyüdüğünde, x ile x'e en yakın sayılar arasındaki mesafelerin de arttığını gözlemliyoruz:

- 1 noktası ile 1'e en yakın sayı arasındaki uzaklık $2.2204E-016$ iken

x	$\lvert x - x_f \rvert$
1×10^{-2}	$1.7347E-018$
1	$2.2204E-016$
1×10^{2}	$1.4211E-014$
1×10^{4}	$1.8190E-012$
1×10^{8}	$1.4901E-008$
1×10^{16}	2
1×10^{32}	$1.8014E+016$

Tablo 2.3: Bilgisayar sayıları ve komşuları arasındaki uzaklık

- 1×10^{16} ile $1 \times 10^{16} + 1$ sayısı arasında hiç bir bilgisayar sayısı yoktur, çünkü en yakın sayı iki birim uzaklıktadır.

  Sıfır komşuluğundaki sayıların birbirlerine çok yakın olması ve mutlak değerce büyük olan kayan nokta sayıları arasındaki uzaklığın kısmen daha büyük olması sonucu sayıların küçükten büyüğe veya büyükten küçüğe sıralanarak toplanması sonucu farklı sonuçlar elde edilebilmektedir. Örneğin

$$\sum_{i=1}^{N} \frac{1}{i^2} = 1 + 1/2^2 + \cdots + 1/N^2$$

  büyükten küçüğe toplamını gözönüne alalım: $N = 1e7$ için

```
top=0;
for i=1:N
   top=top+1/(i*i);
end
```

  program parçacığı çalıştırıldığında

$$top = 1.64492406684726$$

  elde ederiz.

  Şimdi de

$$\sum_{i=1}^{N} \frac{1}{(N-(i-1))^2} = 1/N^2 + 1/(N-1)^2 + \cdots + 1/1^2$$

  olarak ifade edilen küçükten büyüğe toplamını hesaplayalım:

```
top=0;
for i=N:-1:1
 top=top+1/(i*i);
end
```

program parçacığı çalıştırıldığında

$$top \ = 1.64492406684823$$

elde ederiz. Sonuçların farklı olduğunu görüyoruz.

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \pi^2/6 = 1.64493406684823$$

olduğu dikkate alınırsa küçükten büyüğe toplamın doğru sonucu ürettiğini gözlemleriz.

- Büyüklükleri çok farklı sayılarla gerçekleştirilen işlemlerde büyük hatalar oluşabilir. Bu duruma örnek olarak aşağıdaki işlem sonuçlarını inceleyelim:

İşlem	Yaklaşık hata
$0 + 5E-324 = 4.940656458412465E-324$	$6E-326$
$1E5 \times (1 - 1E-17) = 1E5$	$1E-12$
$1E10 \times (1E16 + 1E-1) = 1E26$	$1E+09$

Tablo 2.4: Farklı mertebeden büyüklüklere sahip sayılarla işlemler ve oluşan hata

Gözlem 2.2. *Sonuç olarak büyüklük mertebeleri çok farklı olan bilgisayar sayıları ile yapılan işlemlerde büyük hatalar oluşabilmektedir ve mümkünse işlemlerde çok küçük ve çok büyük sayıların işlemini gerektiren bu tür sonuçların ortaya çıkmaması için alternatif formülasyonlar uygulanmalıdır.*

  - On tabanlı sisteme benzer olarak iki tabanlı sistemde oluşabilecek en büyük bağıl hatanın üst sınırı ise

$$\epsilon_{\max} = \frac{1}{2} \times 2^{-n} = 2^{-(n+1)}$$

olarak elde edilir.

– Öte yandan bilgisayar epsilonu, 1 sayısının komşuluğundaki bir sayının 1 e yuvarlanması sonucu oluşabilen maximum bağıl hata olarak ta tanımlanır. O halde bilgisayar epsilonu 1 ile 1 e en yakın bilgisayar sayısı arasındaki uzaklığın yarısına eşit olmalıdır.

– MATLAB/Octave ortamında $eps()$ fonksiyonu, x_f ile $\overrightarrow{x}_f$ (bir sonraki komşu kayan nokta) arasındaki maksimum bağıl uzaklığı, yani

$$\left| \frac{\overrightarrow{x}_f - x_f}{x_f} \right|$$

oranını verir. $x_f = 1$ olması durumunda bu uzaklık mutlak uzaklığa dönüşür ve bu durumda yukarıdaki tanımda verilen bilgisayar epsilonu

$$\epsilon_{\max} = \frac{1}{2} \times 2^{-n} = \frac{1}{2} eps(1) \tag{2.8}$$

bağıntısından elde edilebilir. Kullandığımız bilgisayar için

>> eps(1)
ans =
$2.2204e-016 (= 2^{-52})$ dir. O halde

$$\epsilon_{\max} = \frac{1}{2} 2^{-52} = 2^{-53}$$

olarak elde edilir. Bu örnekteki

$$p = n + 1 = 53$$

sayısı kayan nokta sayı sisteminin *hassasiyeti* olarak tanımlanır.

2.4.4 $R_f(10) -> R_f(2)$ dönüşüm kaynaklı hatalar

On tabanlı sistemde sonlu basamakla temsil edilebilen bir sayı ikili sistemde sonlu sayıda basamakla temsil edilemeyebilir. Literatürde genelde 0.1 sayısı üzerinden incelenen bu konu diğer bazı tipik ondalıklı sayılar için de geçerlidir[1]. Aşağıdaki tabloda sonlu basamaklı bazı on tabanlı sayıların ikili sistemdeki gösterimleri sunulmaktadır.Tablodaki dönüşümler Program 2.3 yardımıyla hesaplanmıştır. Program sayının 32 bir formatında ikili sistem karşılığını belirledikten sonra, elde edilen ikili sayıyı tekrar on tabanlı sisteme dönüştürerek sonucu kontrol etmektedir.

$0.1 \rightarrow$	$0.000\overline{1100}$	$0.6 \rightarrow$	$0.100\overline{1100}$
$0.2 \rightarrow$	$0.00\overline{1100}$	$0.7 \rightarrow$	$0.10\overline{1100}$
$0.3 \rightarrow$	$0.01\overline{0011}$	$0.8 \rightarrow$	$0.\overline{1100}$
$0.4 \rightarrow$	$0.0\overline{1100}$	$0.9 \rightarrow$	$0.1\overline{1100}$
$0.5 \rightarrow$	0.1		

Tabloda

$$\overline{1100} = 110011001100...$$

gösterimi ile 1100 rakamlar grubunun devrettiği ifade edilmektedir.

Tablodan örneğin $0.1 = 0.000\overline{1100}$ devirli sayısıdır. Ancak 32 bit formatta bu sayının mantisi için 23 bitlik bir alan tahsis edildiği için bu formattaki normalize edilmiş gösterimi

$$0.1 \cong (1.10011001100110011001100)_2 \times 2^{-4}$$

dir. Bu sayının on tabanlı sistemdeki karşılığı ise, virgülden sonra onbeş basamak yuvarlatılmış formatta

$$0.099999904632568$$

olarak elde edilir.

Böylece 0.1 sayısının ikili sistemdeki devirli gösterimi, tahsis edilen bellek alanına yerleştirilemeyerek gösterimde hata oluşmasına neden olunmuştur.

Benzer biçimde $0.2 = 0.00\overline{1100}$ devirli sayısıdır. Ancak 32 bit formatta bu sayının mantisi için 23 bitlik bir alan tahsis edildiği için bu formattaki normalize edilmiş gösterimi

$$0.2 \cong (1.10011001100110011001100)_2 \times 2^{-3}$$

tür. Bu sayının on tabanlı sistemdeki karşılığı ise

$$1.9999996E - 001$$

olarak elde edilir.

Böylece 0.2 sayısının da ikili sistemdeki devirli gösterimi, tahsis edilen bellek alanına yerleştirilemeyerek gösterimde hata oluşmasına neden olunmuştur.

O halde yukarıdaki tabloda ikili tabanda devirli gösterime sahip her sayı ile gerçekleştirilen işlemlerde çok özel durumlar bile olsalar, zaman zaman

sorun yaşanma ihtimali vardır. Bu noktayı vurgulamak üzere aşağıdaki örneği inceleyelim:

$>> toplam = 0;\ for\ i = 1 : 10\ toplam = toplam + 0.2; end$

$>> z = (2 - toplam) * 1E16$

z = 2.220446049250313

elde ederiz. Yukarıdaki örnekten on adet 0.2 nin toplamının 2 olmasını bekleriz. Esasen

$>> toplam$

toplam =

2.0000

olduğu görülmektedir, ancak bu sonuç yuvarlatılarak ekrana yansıtılan değerdir, gerçek sonuç 2 ye eşit değildir. Bu durumda çok küçük bir değer olan $(2 - toplam)$ nin $1E16$ gibi çok büyük bir sayı ile çarpımı sonucunda, normalde sıfır olması beklenen z değeri yukarıda belirtildiği gibi hatalı olarak elde edilmiştir. Benzer durum 0.5 hariç, tablodaki diğer sayıları için de geçerlidir.

2.5 Anlamlı basamak kaybı hatası

Yukarıdaki örnekte taban dönüşümü sonucu elde edilen birbirine çok yakın iki sayının farkının hesaplanmasında oluşan hata, anlamlı basamak kaybı sonucunda ortaya çıkmıştır. Öncelikle anlamlı basamak sayısını tanımlayalım:

TANIM 2.6. *On tabanlı sistemde eğer* $|\varepsilon_b(x)| \leq 5 \times 10^{-k}$ *ise* x_f *yaklaşımı* x *gerçek değerini* $k-1$ *anlamlı basamakla temsil etmektedir denir.*

Aşağıdaki örnekleri inceleyelim:

- $x_f = 1.1$ yaklaşımı $x = 1$ değerini 0 anlamlı basamakla temsil etmektedir, çünkü

$$|\varepsilon_b(x)| = \frac{1.1 - 1}{1} = 1 \times 10^{-1} < 5 \times 10^{-1}$$

dir.

- $x_f = 1.23$ yaklaşımı $x = 1.2$ değerini 1 anlamlı basamkla temsil etmektedir, çünkü

$$|\varepsilon_b(x)| = \frac{1.23 - 1.2}{1.2} = 2.5 \times 10^{-2} < 5 \times 10^{-2}$$

dir.

- $x_f = 1.234$ yaklaşımı $x = 1.232$ değerini 2 anlamlı basamakla temsil etmektedir, çünkü

$$|\varepsilon_b(x)| = \frac{1.234 - 1.232}{1.232} \cong 1.6 \times 10^{-3} \leq 5 \times 10^{-3}$$

tür.

Anlamlı basamak kaybı sonucu oluşan hatayı aşağıdaki örnek yardımıyla daha yakından inceleyelim.

ÖRNEK 2.6. *Cebirsel olarak birbirine eşit olan*

$$h(x) = 1000(log(x+1) - log(x))$$

ve

$$g(x) = 1000log((x+1)/x)$$

fonksiyonlarını göz önüne alarak, bu fonksiyonların $x = 100000$ noktasındaki değerlerini, virgülden sonra 6 basamağa kadar ve yuvarlama prensibine göre çalışan hesaplayıcı ile elde edeceğiniz sonuçları karşılaştırınız.

Çözüm.

x_f ile x'in hesaplayıcıda temsil edilen yaklaşımını gösterelim. Bu durumda

$$\begin{aligned} x &= 100000 \text{ ve} \\ y_1 &= log(x+1) \\ &= 5.00000434292310, \\ y_{1f} &\doteq 5.000004 \end{aligned}$$

tür. y_{1f} değeri y_1 değerini 6 anlamlı basamakla temsil eder, çünkü

$$\begin{aligned} \left|\frac{y_{1f} - y_1}{y_1}\right| &= \left|\frac{5.000004 - 5.00000434292310}{5.00000434292310}\right| \\ &= 6.8585 \times 10^{-8} < 5 \times 10^{-7} \end{aligned}$$

dir. Öteyandan

$$y_2 = y_{2f} = log(x) = 5;$$

dir. O halde

$$h(100000) = 1000(y_{1f} - y_{2f}) = 0.004;$$

ve

$$\begin{aligned} g(100000) &= 1000log((10001)/10000) \\ &\doteq 1000 \times 0.000043 = 0.004343. \end{aligned}$$

olarak elde edilir. Gerçek sonuç ise 0.00434292310448164 dir. y_{1f} değeri 6 anlamlı basamağa sahipken birbirine çok yakın y_{1f} ve y_{2f} sayılarının farkı

$$y_{1f} - y_{2f} = 0.000004$$

olarak elde edilir. $y_{1f} - y_{2f}$ yaklaşımı,

$$y_1 - y_2 = 0.000004342923104 69431$$

gerçek değerini *hiçbir anlamlı basamakla temsil etmez*, çünkü

$$\begin{aligned} & \left| \frac{(y_{1f} - y_{2f}) - (y_1 - y_2)}{(y_1 - y_2)} \right| \\ = & \left| \frac{0.000004 - 0.000004342923104 69431}{0.000004342923104 69431} \right| \\ = & \ 7.896\,13576910079 \times 10^{-2} < 5 \times 10^{-1} \end{aligned}$$

olup, anlamlı basamak sayısı tanımında $k = 1$ dir. Dolayısıyla fark işlemi anlam basamağı kaybına neden olmuştur. Söz konusu basamak kaybı, birbirine eşit olan h ve g fonksiyonları yardımıyla elde edilen sonuçların farklılık göstermesine neden olmuştur. g ile elde edilen sonucun gerçek sonuca çok yakın iken, f ile elde edilen sonucun ise gerçek sonuçtan çok farklı olduğunu gözlemliyoruz.

O halde birbirine yakın değerler alması muhtemel değişkenlerin farklarının alınmasından kaçınmak gerekmektedir. Bunun için yukarıdaki örnekte olduğu üzere, verilen ifade belirtilen değişken değerlerinin farkının hesaplanmasını içermeyen alternatif biçimlerde yazılmalıdır. Örneğin aşağıdaki tabloda sol tarafta yer alan fonksiyon yazılımları yerine belirtilen x komşuluğunda sağ sütunda yer alan denk formülasyonlar kullanılmalıdır.

Benzer biçimde mutlak değerce küçük sayıların bölümü de kaçınılması gereken bir diğer işlem türüdür.

$f(x)$	$\rightarrow$	$f(x)$
$ln(x) - ln(y)$	$x \simeq y$	$ln(x/y)$
$1 - 1/(1+x)$	$x \simeq 0$	$x/(1+x)$
$1 - \cos(x)$	$x \simeq 0$	$\sin^2(x)/(1+\cos(x))$

Alıştırmalar 2.1.

1. *Aşağıda verilen saylar için*

   (a) kesme prensibine göre kayan noktadan sonra 4 basamaklı ve

   (b) yuvarlama prensibine göre de 4 basamaklı yaklaşımların kullanılması durumunda oluşacak olan mutlak ve bağıl hataları hesaplayınız.

   - $x = 3.14159265358979$
   - $x = 2.71828182845905$
   - $x = 1.41421356237310$
   - $x = 1.73205080756888$

2. $$x_f = \pm q_f \times 10^e,\ q_f = d_0.d_1d_2,\ d_{0,1,2} = 0, 1, \ldots, 5\ ; d_0 \neq 0$$

   ve

   $$e = -1, 0, 1$$

   üs değerlerine sahip bir R_f sayı sistemi gözönüne alalım ve gerçekleştirilen her işlem sonucunun R_f de temsil edilebilen aralıkta olup, ancak R_f de yer almaması durumunda, sistemde bulunan en yakın sayıya yuvarlatıldığını kabul edelim. Buna göre

   - *R_f in elemanları(sayıları) nelerdir?*
   - *R_f in en büyük pozitif ve en küçük pozitif sayıları nelerdir?*
   - *R_f kümesinin toplama işlemine göre kapalılık özelliğini sağlamadığını bir örnekle gösteriniz.*
   - *R_f de toplama işlemine göre birleşme özelliği olmadığını bir örnekle gösteriniz.*
   - *R_f deki bağıl yuvarlama hatalarının üst sınırını temsil eden bilgisayar epsilonu nedir?*

- R_f *de inf ve NaN gösterimlerine sahip olabilecek birer işlem tanımlayınız*
- R_f *deki subnormal sayılar kümesini belirleyiniz.*

3. *Aşağıda verilen on tabanlı sayıların karşılarında verilen iki tabanlı gösterimlere sahip olduklarını gösteriniz*

$3 \rightarrow$	$(11)_2$	$50 \rightarrow$	$(110010)_2$
$8 \rightarrow$	$(1000)_2$	$100 \rightarrow$	
$10 \rightarrow$	$(1010)_2$	$200 \rightarrow$	$(11001000)_2$
$20 \rightarrow$	$(10100)_2$	$500 \rightarrow$	$(111110100)_2$
$40 \rightarrow$	$(101000)_2$	$1000 \rightarrow$	$(1111101000)_2$

4. *Aşağıda verilen on tabanlı sayıların iki tabanlı gösterimlerini belirleyiniz.*

- 25.1
- 33.25
- 75.454
- 96.2456

5. *Soru 3 de elde ettiğiniz iki tabanlı gösterimleri on tabanlı sisteme dönüştürerek, başlangıçtaki on tabanlı sayıları elde ettiğinizi kontrol ediniz.*

6. *Aşağıda verilen iki tabanlı sayıların on tabanlı gösterimlerini belirleyiniz. Sonuçlarınızı Program 2.1 ile kontrol ediniz.*

- $(11001)_2$
- $(1001011)_2$
- $(1100000.101)_2$
- $(100001.1101)_2$

7. *Soru 6 da verilen iki tabanlı sayıların $R_f(2) - 32$ bit bellek gösterimlerini belirleyiniz.*

8. *Soru 6 da verilen iki tabanlı sayıların $R_f(2) - 64$ bit bellek gösterimlerini belirleyiniz.*

9. *Aşağıda verilen kesirli sayıların iki tabanlı gösterimlerinin doğruluğunu kontrol ediniz.*

   - $2.5 \rightarrow (10.1)_2$
   - $1.25 \rightarrow (1.01)_2$
   - $1.125 \rightarrow (1.001)_2$
   - $1.3125 \rightarrow (1.0101)_2$

10. *Aşağıda verilen kesirli sayıların iki tabanlı gösterimlerinin doğruluğunu kontrol ediniz.*$\overline{1001}$ *üs çizgi notasyonu sayının devirli sayı olduğunu ifade etmektedir.*

    - $0.1 \rightarrow (0.0\overline{0011})_2$
    - $3.6 \rightarrow (1.1\overline{1001})_2$

11. *Bilgisayar sisteminizde* $[1, 2)$ *aralığındaki kayan nokta sayıları arasındaki uzaklığın birbirine eşit olduğunu MATLAB/Octave* eps *komutu yardımıyla gözlemleyiniz. Buna göre örneğin* $eps(1) = eps(1.5) = eps(1.8)$ *olmalıdır.*

12. *Bilgisayar sisteminizde* $[2^n, 2^{n+1})$ *aralığında* $(\mathsf{n} = 0, 1, 2, ...)$ *yer alan sayılar arasındaki uzaklıkların birbirine eşit olduğunu ve her bir aralıktaki sayılar arasındaki mesafenin bir önceki aralıktaki sayılar arasındaki mesafenin* 2 *katı olduğunu gözlemleyiniz.*

13. *Bilgisayar sisteminizdeki pozitif ve en küçük subnormal sayıyı belirleyiniz. Elde ettiğiniz sayıdan daha küçük ve negatif olmayan tek sayı sıfır olmalıdır.*

14. *Bilgisayarınızda temsil edilebilecek en büyük sayının 64-bit formatta* $(2 - eps) \times 2^{1023}$ *olduğunu MATLAB/Octave ortamında gözlemleyiniz. Bu durumda* 2^{1024} *sayı sisteminizde yer alır mı?*

15. *Aşağıda verilen yaklaşımların karşılarında verilen değerleri kaç anlamlı basamakla temsil ettiğini açıklayınız*

    - $x_f = 12.36$, $x = 12.345$
    - $x_f = 0.0013$, $x = 0.00125$
    - $x_f = 0.000011$, $x = 0.000012$

- $x_f = 3.14159, x = 3.141592653589793$

16. *Anlam basamak kaybı hatasının önlenmesi için aşağıdaki fonksiyonlar belirtilen nokta komşuluğunda alternatif olarak nasıl yazılabilir?*

    - $e^{-x}, x > 0$
    - $(-b + \sqrt{b^2 - 4c})/2, b >> 0, c \simeq 0$(*burada* $>>$ *sembolü çok büyük anlamındadır,* $a \simeq b$ *gösterimi ise* a *nın* b *ye çok yakın olduğunu ifade etmektedir).*
    - $e^{x-y}, x \simeq y$
    - $\sqrt{x+1} - \sqrt{x}, x >> 1$

17. *Program 2.4 ü MATLAB/ OCTAVE ortamında çalıştırarak, yukarıda elde ettiğiniz taban dönüşümlerinin doğruluğunu kontrol ediniz.*

Kaynaklar

[1] Atkinson, K. An Introduction to Numerical Analysis, John Wiley & Sons, 1988.

[2] Coşkun, E. Octave ile Sayısal Hesaplama ve Kodlama(URL:aves.ktu.edu.tr/erhan/dokumanlar).

[3] Kincaid, D., Cheney, W., Numerical Analysis, Brooks/Cole, 1991.

[4] Mathews, J., Numerical Methods for Mathematics, Science and Engineering, Prentice-Hall, 1997.

[5] MATLAB, Mathworks(*URL:mathworks.com).*

[6] Octave, GNU özgür yazılım(*URL:Octave.sourceforge.net*).

[7] Overton, M. L., Numerical Computing with IEEE Floating Point Arithmetics, SIAM, New York, 2001.

[8] M., Cleve, Floating point Numbers, (URL:blogs.mathworks.com/cleve/2014/07/07/floating-point-numbers).

[9] Press, H. W. ve ark., Numerical Recipes in C, Cambridge University Press, 1988.

[10] Stoer, J., Bulirsh, R., Introduction to Numerical Analsis, Springer-Verlag, 1976.