Desimal sistemde reel sayıların, tüm rakamlarının desimal noktanın sağında kalacak şekilde ve noktadan sonraki ilk rakam sıfırdan farklı olacak şekilde yazıldığı gösterime **normalleştirilmiş bilimsel gösterim** denir.

Örnek:

$$\begin{aligned} 732.5051 &= 0.7325051 \times 10^{3} \\ -0.005612 &= -0.5612 \times 10^{-2} \end{aligned}$$

Genel olarak r, $\frac{1}{10} \leq r < 1$ aralığında bir sayı ve n de bir tamsayı (pozitif, negatif veya sıfır) olmak üzere, sıfırdan farklı bir x sayısı

$$x = \pm r \times 10^n$$

formunda temsil edilebilir. Kuşkusuz, eğer $x = 0$ ise, bu durumda $r = 0$ olup, diğer tüm durumlarda, r verilen aralıkta kalacak şekilde n yi ayarlayabiliriz.

Tamamen aynı yolla, bilimsel gösterimi **ikilik sistem** için de kullanabiliriz. O durumda ($x \neq 0$ ise) $\frac{1}{2} \leq q < 1$ ve m bir tamsayı olmak üzere,

$$x = \pm q \times 2^m \tag{4}$$

dir.
q sayısı **mantissa** (anlamlı rakamlar kısmı) ve m tamsayısı da **üs** olarak adlandırılır. q ve m nin her ikisi de 2-tabanlı sayılardır.

İkili sayıları bir bilgisayara yüklerken, (4) formunda küçük bir düzenleme yapılması yerinde olur: Baştaki ikilik rakam 1 in ikilik-noktanın hemen sağına kaydırıldığını varsayalım. Bu durumda gösterim $q = (1.\mathrm{f})_2$ ve $1 \leq q < 2$ şeklinde olacaktır. Böylece, baştaki 1 i oradaymış gibi düşünüp, fakat gerçekte yüklemeden bir bitlik bir alan kazanılarak, sadece $(.\mathrm{f})_2$ bir bilgisayar kelimesine yüklenecektir.

$32 - bit$ lik bir bilgisayarda çalıştığımızı varsayacağız.
Böyle bir bilgisayarda, bir kelimeyi oluşturan bitler, sıfırdan farklı bir $x = \pm q \times 2^m$ reel sayısının temsilinde aşağıdaki şekilde düzenlenir:

x reel sayısının işareti	1 bit
üs kuvveti (e tamsayısı)	8 bit
mantissa kısmı (f reel sayısı)	23 bit

$x = \pm q \times 2^m$ reel sayısı **sol-kaymalı normalleştirilmiş ikili sayı** olarak yazılabilir öyle ki mantissadaki sıfırdan farklı ilk bit ikilik noktanın hemen önündedir. Yani $q = (1.\mathrm{f})_2$ dir. Bu bit her zaman 1 kabul edildiğinden, yüklemeye gerek kalmaz. Mantissa $1 \leq q < 2$ aralığındadır. Kelimede mantissa için ayrılan 23 bit, f den 23 bit yüklemek için kullanılabilir. Bunun anlamı ise, makinenin kayan-noktalı sayıları için 24 bitlik mantissaya sahip olacağıdır.

Böylece, sıfırdan farklı normalleştirilmiş makine sayıları, değerleri aşağıdaki gibi yeniden kodlanan bit alanlarıdır:

$$q = (1.\mathrm{f})_2 \text{ ve } m = \mathrm{e} - 127$$

olmak üzere

$$x = (-1)^s q \times 2^m \tag{5}$$

dir. Burada, $1 \leq q < 2$ ve q daki en anlamlı bit 1 olup, açık olarak yüklenmez. Ayrıca, s de x in işaretini (pozitif: bit 0, negatif: bit 1) temsil eden bittir. $m = e - 127$, 8-bitlik üs ve f de x reel sayısının 23- bitlik kesirli kısmıdır ki baştaki açık 1 bit ile anlamlı rakam alanı $(1.\square\square\square\cdots\square\square\square)_2$ yi verir.

Kayan-noktalı sayılar ve yuvarlama hataları

Eğer sayı, $|m|$ 8-bitlik ve q 23-bitlik yer işgal edecek şekilde temsil ediliyorsa, bu `32-bitlik` bir bilgisayarda bir **makine sayısıdır**.
Yani bu sayı duyarlı olarak temsil edilebilir.
Çoğu sayının `32-bitlik` bir bilgisayarda duyarlı temsili yoktur.
Eğer böyle bir sayı makineye bir girdi olarak veriliyorsa veya makinede bir hesap sonucu oluşuyorsa, onu bir **makine sayısı** olarak duyarlı bir şekilde temsil etmede kaçınılmaz hatalar oluşur.

Kayan-noktalı sayılar ve yuvarlama hataları

$|m|$ nin 8 bitten fazla olmaması kısıtlamasının anlamı

$$0 < \mathrm{e} < (11\ 111\ 111)_2 = 2^8 - 1 = 255$$

olup, $\mathrm{e} = 0$ ve $\mathrm{e} = 255$ değerleri ± 0, $\pm\infty$ ve `NaN` (Sayı Değil) özel durumları için ayrılmıştır. $m = e - 127$ olduğundan $-126 \leq m \leq 127$ almaktayız ki böylece 32-bit lik bir bilgisayar, $2^{-126} \approx 1.2 \times 10^{-38}$ e kadar küçük ve $(2 - 2^{-23})2^{127} \approx 3.4 \times 10^{38}$ e kadar büyük olan sayıları işleyebilir.
Bu ise bilimsel hesaplamalarda yeterince geniş bir alan değildir.

Kayan-noktalı sayılar ve yuvarlama hataları

Bu nedenle, bilimsel hesaplamalarda programlar bazen **çift-duyarlı** veya **genişletilmiş-duyarlılık aritmetiğiyle** yazılmak zorunda kalınır.
Çift-duyarlı bir kayan-noktalı sayı iki bilgisayar kelimesiyle temsil edilir ve mantissası çoğu zaman en az iki kat fazla bite sahiptir. Böylece, çift duyarlılıkta tek duyarlılığın en az iki katı sayıda desimal nokta duyarlılığına sahip oluruz. Çift duyarlılıkta hesaplamalar tek duyarlılığa göre, çoğu zaman 2 veya katı, daha yavaştır. Bunun nedeni, tek duyarlı aritmetik donanım tarafından yapılırken, çift duyarlı aritmetiğin genellikle bir yazılım aracılığıyla yapılmasındandır.

Kayan-noktalı sayılar ve yuvarlama hataları

Makine sayılarının mantissa kısmı 23 bitten fazla olmamalıdır.
Anlamlı en küçük bit

$$2^{-23}$$

ü temsil eder.

$$2^{-23} \approx 1.2 * 10^{-7}$$

dir.
Yani makine sayıları kabaca altı desimal duyarlılığa sahip olabilir.

Kayan-noktalı sayılar ve yuvarlama hataları

- Bilgisayara altı desimal noktadan daha fazlasıyla temsil edilen bir sayı girildiğinde onun yaklaşık değeri alınır.

Kayan-noktalı sayılar ve yuvarlama hataları

- Bilgisayara altı desimal noktadan daha fazlasıyla temsil edilen bir sayı girildiğinde onun yaklaşık değeri alınır.
- Ayrıca, $1/100$ gibi bazı *basit* desimal sayılar da bir ikili sistem bilgisayarında makine sayısı değildirler!

Kayan-noktalı sayılar ve yuvarlama hataları

Bir **tamsayı**, işaret için ayrılması gereken tek bit haricinde, bilgisayar kelimesinin tümünü kendi temsili için kullanabilir. Böylece, 32-bitlik bir bilgisayarda tamsayılar $-(2^{31}-1)$ ile $2^{31}-1 = 21474\ 83647$ aralığında değişirler.
Bilimsel hesaplamalarda, tamamıyla tamsayısı içeren işlemlerle çok sık karşılaşılmaz.

Sıfır, Sonsuz ve NaN

IEEE standart aritmetiğinde 0 ın iki formu, $+0$ ve -0 mevcut olup, tek duyarlılıkta sırası ile $[00000000]_{16}$ ve $[80000000]_{16}$ bilgisayar kelimeleriyle temsil edilirler. Bir 0 değer ile sonuçlanan çoğu aritmetik işlemine $+0$ değeri verilir. Makine duyarlılığında 0 üreten çok minik bir negatif sayıya ise -0 değeri verilir.

Sıfır, Sonsuz ve NaN

Benzer olarak sonsuzun da iki formu, $+\infty$ ve $-\infty$ var olup, tek duyarlılıkta sırası ile $[7F800000]_{16}$ ve $[FF800000]_{16}$ bilgisayar kelimeleriyle temsil edilirler. Sonsuzluk, çoğunlukla, anlamlı işlendiği her durumda, çok büyük bir sayı olarak ele alınır. Örneğin, x i $0 < x < \infty$ aralığında bir kayan-noktalı sayı olarak kabul edersek, bu durumda x/∞ işlemi $+0$ değerini alırken, $x + \infty, x * \infty$ ve ∞/x işlemlerinin hepsine $+\infty$ değeri atanır. Burada ∞ dan anlaşılan $+\infty$ dur. $-\infty$ için de benzer sonuçlar geçerlidir.

Sıfır, Sonsuz ve NaN

`NaN` ın anlamı **Sayı Değil** (Not a Number) dir ve $0/0, \infty - \infty, x+$`NaN` gibi belirsiz operasyonlar sonucunda oluşur. `NaN`, e $= 255$ ve f $\neq 0$ olan bilgisayar kelimesi ile temsil edilir.

Yakın Makine Sayıları

Verilen bir pozitif x reel sayısına, yakın makine sayısı ile yaklaşmanın sonucu ortaya çıkan hatayı inceleyelim:

$$x = q \times 2^m \qquad 1 \leq q < 2 \qquad -126 \leq m \leq 127$$

kabul edelim. x e en yakın makine sayısının ne olduğunu bulacağız.

Yakın Makine Sayıları

a_i ler 0 veya 1 olmak üzere,

$$x = (1.a_1 a_2 \ldots a_{23} a_{24} a_{25} \ldots)_2 \times 2^m$$

yazalım.
Yakın makine sayısı basitçe $a_{24} a_{25} \ldots$ uzantı bitlerini atarak elde edilebilir.
Bu yordama genellikle **yutma** denir.
Sonuç,

$$x_- = (1.a_1 a_2 \ldots a_{23})_2 \times 2^m$$

dir.
Bu sayı reel eksende x in solunda kalır.

Yakın Makine Sayıları

x e yakın olan bir başka sayı da onun sağında kalır.
Bu sayı **yuvarlama** ile elde edilir; yani uzantı bitlerini önceki gibi atarak, fakat en son kalan a_{23} bitini bir birim artırarak elde edilir.

$$x_+ = \left((1.a_1a_2...a_{23})_2 + 2^{-23}\right) \times 2^m$$

x_- ve x_+ nın x e daha yakın olanı bilgisayarda x i temsil etmek için seçilir.

Yakın Makine Sayıları

Eğer x, x_- ile daha iyi temsil ediliyorsa,

$$|x - x_-| \leq \tfrac{1}{2} |x_+ - x_-| = \tfrac{1}{2} \times 2^{m-23} = 2^{m-24}$$

olur.
Bu durumda **bağıl hata** aşağıdaki şekilde sınırlıdır:

$$\left| \frac{x - x_-}{x} \right| \leq \frac{2^{m-24}}{q \times 2^m} = \frac{1}{q} \times 2^{-24} \leq 2^{-24}$$

Yakın Makine Sayıları

İkinci durumda, x, x_+ ya x_- den daha yakın olup,

$$|x - x_+| \leq \frac{1}{2} |x_+ - x_-| = 2^{m-24}$$

dür.
Aynı analiz bağıl hatanın 2^{-24} den daha büyük olamayacağını gösterir.

Yakın Makine Sayıları

Özetlersek;
eğer x, makinenin tanım bölgesinde kalan, sıfırdan farklı bir reel sayıysa, bu durumda x e en yakın x^* makine sayısı

$$\left|\frac{x - x^*}{x}\right| \leq 2^{-24}$$

eşitsizliğini sağlar. $\delta = (x^* - x)/x$ dersek, bu eşitsizliği

$$\mathrm{fl}(x) = x(1+\delta) \qquad |\delta| \leq 2^{-24} \tag{6}$$

formunda yazabiliriz.
$\mathrm{fl}(x)$ notasyonu, x e en yakın olan x^* kayan-noktalı makine sayısını belirtmek için kullanılır.

Yakın Makine Sayıları

Örnek

ÖRNEK 1 $x = 2/3$ sayısının ikilik formu nedir? 32-bitlik bir makinede x_- ve x_+ yakın makine sayıları nelerdir? Bunların hangisi $\mathrm{fl}(x)$ olarak alınmalıdır? x i $\mathrm{fl}(x)$ ile temsil edersek, mutlak yuvarlama hatası ve bağıl yuvarlama hatası ne olur?

Yakın Makine Sayıları

Çözüm

$$x = \frac{2}{3} = (0.1010...)_2 = (1.010101...)_2 \times 2^{-1}$$

İki yakın makine sayısı

$$\begin{aligned} x_- &= (1.0101...010)_2 \times 2^{-1} \\ x_+ &= (1.0101...011)_2 \times 2^{-1} \end{aligned}$$

Burada, x_-, yutma ile ve x_+ da yuvarlama ile elde edilmiştir. İkilik noktanın sağında 23 bit vardır.

Yakın Makine Sayıları

Çözüm

$$\begin{aligned} x - x_- &= (0.1010...)_2 \times 2^{-24} = \tfrac{2}{3} \times 2^{-24} \\ x_+ - x &= (x_+ - x_-) - (x - x_-) = 2^{-24} - \tfrac{2}{3} \times 2^{-24} = \tfrac{1}{3} \times 2^{-24} \end{aligned}$$

olduğundan, $\mathrm{fl}(x) = x_+$ almalıyız.

Bu durumda, mutlak yuvarlama hatası

$$|\mathrm{fl}(x) - x| = \tfrac{1}{3} \times 2^{-24}$$

bağıl yuvarlama hatası

$$\frac{|\mathrm{fl}(x) - x|}{|x|} = \frac{\frac{1}{3} \times 2^{-24}}{\frac{2}{3}} = 2^{-25}$$

olur. Çıktı alındığında ortaya çıkan sayılar ise

$$\begin{aligned} x_- &= [0011\ 1111\ 0010\ 1010\ 1010\ 1010\ 1010\ 1010]_2 = [3\mathrm{F2AAAAA}]_{16} \\ x_+ &= [0011\ 1111\ 0010\ 1010\ 1010\ 1010\ 1010\ 1011]_2 = [3\mathrm{F2AAAAB}]_{16} \end{aligned}$$

$$x_- = [0011\ 1111\ 0010\ 1010\ 1010\ 1010\ 1010\ 1010]_2 = [3F2AAAAA]_{16}$$
$$x_+ = [0011\ 1111\ 0010\ 1010\ 1010\ 1010\ 1010\ 1011]_2 = [3F2AAAAB]_{16}$$

$$x_- = 0.66666\ 66269\ 30236\ 81640\ 62500\ 000$$
$$x_+ = 0.66666\ 66865\ 34881\ 59179\ 68750\ 000$$

Burada ikisi arasındaki mutlak boşluk
$0.00000\ 00596\ 04644\ 77539\ 06250\ 000 = 2^{-24}$ dür.