17. IEEE floating point numbers

- floating point numbers with base 10
- floating point numbers with base 2
- IEEE floating point standard
- machine precision
- rounding error

Floating point numbers with base 10

$$x = \pm\,(.d_1 d_2 \ldots d_n)_{10} \cdot 10^e$$

- $.d_1 d_2 \ldots d_n$ is the *mantissa* (d_i integer, $0 \le d_i \le 9$, $d_1 \ne 0$ if $x \ne 0$)
- n is the *mantissa length* (or *precision*)
- e is the *exponent* ($e_{\min} \le e \le e_{\max}$)

Interpretation: $x = \pm(d_1 10^{-1} + d_2 10^{-2} + \cdots + d_n 10^{-n}) \cdot 10^e$

Example (with $n = 6$):

$$\begin{aligned} 12.625 &= +\,(.126250)_{10} \cdot 10^2 \\ &= +\,(1 \cdot 10^{-1} + 2 \cdot 10^{-2} + 6 \cdot 10^{-3} + 2 \cdot 10^{-4} \\ &\qquad + 5 \cdot 10^{-5} + 0 \cdot 10^{-6}) \cdot 10^2 \end{aligned}$$

used in pocket calculators

Properties

- a finite set of numbers
- unevenly spaced: distance between floating point numbers varies
  - the smallest number greater than 1 is $1 + 10^{-n+1}$
  - the smallest number greater than 10 is $10 + 10^{-n+2}$, $\ldots$
- largest positive number:

$$+(.999\cdots 9)_{10} \cdot 10^{e_{\max}} = (1 - 10^{-n})10^{e_{\max}}$$

- smallest positive number:

$$x_{\min} = +(.100\cdots 0)_{10} \cdot 10^{e_{\min}} = 10^{e_{\min}-1}$$

Floating point numbers with base 2

$$x = \pm(.d_1 d_2 \ldots d_n)_2 \cdot 2^e$$

- $.d_1 d_2 \ldots d_n$ is the *mantissa* ($d_i \in \{0, 1\}$, $d_1 = 1$ if $x \neq 0$)
- n is the *mantissa length* (or *precision*)
- e is the *exponent* ($e_{\min} \leq e \leq e_{\max}$)

Interpretation: $x = \pm(d_1 2^{-1} + d_2 2^{-2} + \cdots + d_n 2^{-n}) \cdot 2^e$

Example (with $n = 8$):

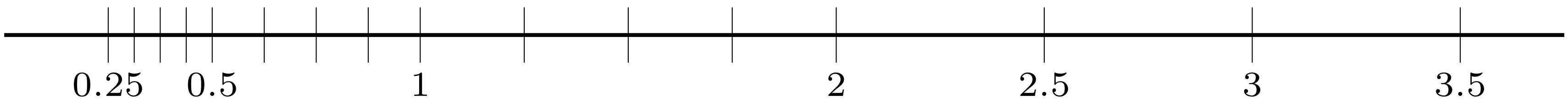
$$\begin{aligned} 12.625 &= +(.11001010)_2 \cdot 2^4 \\ &= +(1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5} \\ &\qquad + 0 \cdot 2^{-6} + 1 \cdot 2^{-7} + 0 \cdot 2^{-8}) \cdot 2^4 \end{aligned}$$

used in almost all computers

Small example

we enumerate all positive floating point numbers for

$$n = 3, \qquad e_{\min} = -1, \qquad e_{\max} = 2$$



$$\begin{array}{ll}
+(.100)_2 \cdot 2^{-1} = 0.2500 & \qquad +(.100)_2 \cdot 2^{0} = 0.500 \\
+(.101)_2 \cdot 2^{-1} = 0.3125 & \qquad +(.101)_2 \cdot 2^{0} = 0.625 \\
+(.110)_2 \cdot 2^{-1} = 0.3750 & \qquad +(.110)_2 \cdot 2^{0} = 0.750 \\
+(.111)_2 \cdot 2^{-1} = 0.4375 & \qquad +(.111)_2 \cdot 2^{0} = 0.875
\end{array}$$

$$\begin{array}{ll}
+(.100)_2 \cdot 2^{1} = 1.00 & \qquad +(.100)_2 \cdot 2^{2} = 2.0 \\
+(.101)_2 \cdot 2^{1} = 1.25 & \qquad +(.101)_2 \cdot 2^{2} = 2.5 \\
+(.110)_2 \cdot 2^{1} = 1.50 & \qquad +(.110)_2 \cdot 2^{2} = 3.0 \\
+(.111)_2 \cdot 2^{1} = 1.75 & \qquad +(.111)_2 \cdot 2^{2} = 3.5
\end{array}$$

Properties

for the floating point number system on page 17-4

- a finite set of unevenly spaced numbers
- the largest positive number is

$$x_{\max} = +(.111\cdots 1)_2 \cdot 2^{e_{\max}} = (1 - 2^{-n})2^{e_{\max}}$$

- the smallest positive number is

$$x_{\min} = +(.100\cdots 0)_2 \cdot 2^{e_{\min}} = 2^{e_{\min}-1}$$

in practice, the number system includes *'subnormal'* numbers

- unnormalized small numbers: $d_1 = 0$, $e = e_{\min}$
- includes the number 0

IEEE standard for binary arithmetic

specifies two binary floating point number formats

IEEE standard single precision

$$n = 24, \qquad e_{\min} = -125, \qquad e_{\max} = 128$$

requires 32 bits: 1 sign bit, 23 bits for mantissa, 8 bits for exponent

IEEE standard double precision

$$n = 53, \qquad e_{\min} = -1021, \qquad e_{\max} = 1024$$

requires 64 bits: 1 sign bit, 52 bits for mantissa, 11 bits for exponent

used in almost all modern computers

Machine precision

Machine precision (of the binary floating point number system on page 17-4)

$$\epsilon_M = 2^{-n}$$

n is the mantissa length

Example: IEEE standard double precision

$$n = 53, \qquad \epsilon_M = 2^{-53} \simeq 1.1102 \cdot 10^{-16}$$

Interpretation

$1 + 2\epsilon_M$ is the smallest floating point number greater than 1:

$$(.10 \cdots 01)_2 \cdot 2^1 = 1 + 2^{1-n} = 1 + 2\epsilon_M$$

Rounding

- a floating-point number system is a finite set of numbers
- all other numbers must be rounded
- we use the notation $\mathrm{fl}(x)$ for the floating-point representation of x

Rounding rules

- numbers are rounded to the nearest floating-point number
- ties are resolved by rounding to the number with least significant bit 0 ('round to nearest even')

Example: numbers $x \in (1, 1 + 2\epsilon_M)$ are rounded to 1 or $1 + 2\epsilon_M$

$$\begin{array}{ll} \mathrm{fl}(x) = 1 & \text{for } 1 \leq x \leq 1 + \epsilon_M \\ \mathrm{fl}(x) = 1 + 2\epsilon_M & \text{for } 1 + \epsilon_M < x \leq 1 + 2\epsilon_M \end{array}$$

therefore numbers between 1 and $1 + \epsilon_M$ are indistinguishable from 1

Rounding error and machine precision

general bound on the rounding error:

$$\frac{|\operatorname{fl}(x) - x|}{|x|} \le \epsilon_M$$

- machine precision gives a bound on the relative error due to rounding
- number of correct (decimal) digits in $\operatorname{fl}(x)$ is roughly

  $$-\log_{10} \epsilon_M$$

  i.e., about 15 or 16 in IEEE double precision
- fundamental limit on accuracy of numerical computations

Exercises

Exercise 1: explain the following results in MATLAB

```
>> (1 + 1e-16) - 1
ans = 0

>> (1 + 2e-16) - 1
ans = 2.2204e-16

>> (1 - 1e-16) - 1
ans = -1.1102e-16

>> 1 + (1e-16 - 1)
ans = 1.1102e-16
```

Exercise 2: run the following code in MATLAB and explain the result

```
x = 2;
for i=1:54
   x = sqrt(x);
end;
for i=1:54
   x = x^2;
end
```

Exercise 3: explain the following results ($\log(1+x)/x \approx 1$ for small x)

```
>>  log(1 + 3e-16) / 3e-16
ans = 0.7401

>>  log(1 + 3e-16) / ((1 + 3e-16) - 1)
ans = 1.0000
```

Exercise 4: the function $f(x) = 1$ for $x \in [10^{-16}, 10^{-15}]$, evaluated as

```
((1 + x) - 1) / (1 + (x - 1))
```