Finite representation of real numbers

Floating-point numbers

Dr. Ing. Rodrigo Gonzalez

rodralez@fing.uncu.edu.ar

Control y Sistemas

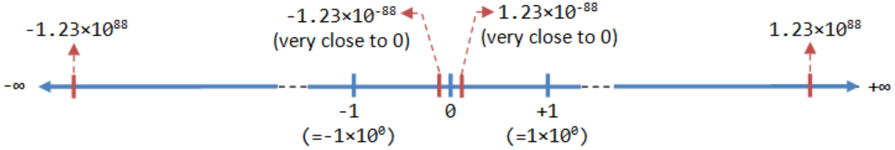
Facultad de Ingeniería,
Universidad Nacional de Cuyo

Summary

1. Floating-point Representation
   - Floating-point Representation
   - Standards
   - Normalized Form
2. Floating-point Examples
3. De-normalized Form
4. Special values
5. Rounding schemes
6. Dynamic range
7. Precision

Floating-point Representation

A floating-point number can represent a very large or a very small value, positive and negative.



Floating-point Numbers (Decimal)

A floating-point number is typically expressed in the scientific notation in the form of

$$(-1)^S \times F \times r^E \,,$$

where,

- S, sign bit.
- F, fraction.
- E, exponent.
- r, certain radix. $r = 2$ for binary; $r = 10$ for decimal.

Standards

Modern computers adopt IEEE 754-2008 standard for representing floating-point numbers.

IEEE Standard P754 Format

Bit	31	30	29	28	27	26	25	24	23	22	21	20	...	2	1	0
	S	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	...	2^{-21}	2^{-22}	2^{-23}
	Sign (s)	← Exponent (e) →								← Fraction (f) →						

IBM Format

Bit	31	30	29	28	27	26	25	24	23	22	21	20	...	2	1	0
	S	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	...	2^{-22}	2^{-23}	2^{-24}
	Sign (s)	← Exponent (e) →							← Fraction (f) →							

DEC (Digital Equipment Corp.) Format

Bit	31	30	29	28	27	26	25	24	23	22	21	20	...	2	1	0
	S	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-2}	2^{-3}	2^{-4}	...	2^{-22}	2^{-23}	2^{-24}
	Sign (s)	← Exponent (e) →								← Fraction (f) →						

MIL-STD 1750A Format

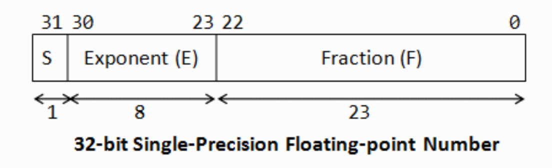
Bit	31	30	29	...	11	10	9	8	7	6	5	4	3	2	1	0
	2^0	2^{-1}	2^{-2}	...	2^{-20}	2^{-21}	2^{-22}	2^{-23}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	← Fraction (f) →								← Exponent (e) →							

IEEE 754-2008 standard

IEEE 754-2008 standard defines several formats.

	Binary formats ($B = 2$)				Decimal formats ($B = 10$)		
Parameter	Binary 16	Binary 32	Binary 64	Binary 128	Decimal 132	Decimal 164	Decimal 128
p, digits	$10 + 1$	$23 + 1$	$52 + 1$	$112 + 1$	7	16	34
e_{max}	+15	+127	+1023	+16383	+96	+384	+16,383
e_{min}	−14	−126	−1022	−16382	−95	−383	−16,382
Common name	Half precision	Single precision	Double precision	Quadruple precision			

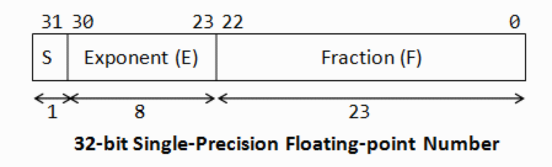
IEEE-754 32-bit Single-Precision



$$(-1)^S \times F \times r^{(E-bias)}$$

- S, sign bit. 0 for positive numbers and 1 for negative numbers.
- E, 8-bits exponent.
- We need to represent both positive and negative exponents.
- $E = [1, 254]$, $bias = 127$; $\quad -126 \leq E - bias \leq 127$.
- $E = 0$ and $E = 255$ are reserved.
- F, 23-bits fraction.

Normalized Form



- Representation of a floating point number may not be unique: $11.01_2 = 1.101_2 \times 2^1 = 110.1_2 \times 2^{-1}$.
- Therefore, the fractional part F is normalized.
- $1.F$, implicit leading 1.

Example 1

Represent 3215.020002_{10}

Decimal Value Entered: 3215.020002

Single precision (32 bits):

Binary: *Status:* normal

Bit 31 Sign Bit	Bits 30 - 23 Exponent Field	Bits 22 - 0 Significand
0	100 0101 0	1 .100 1000 1111 0000 0101 0010
0: + 1: -	Decimal value of exponent field and exponent 138 - 127 = 11	Decimal value of the significand 1.5698340

Hexadecimal: 4548F052 *Decimal:* 3215.0200

http://babbage.cs.qc.cuny.edu/IEEE-754.old/Decimal.html

Example 2

Represent $3215.020002_{10} \times 2 = 6430.040004_{10}$

Decimal Value Entered: 6430.040004

Single precision (32 bits):

Binary: *Status:* normal

Bit 31 Sign Bit	Bits 30 - 23 Exponent Field	Bits 22 - 0 Significand
0	10001011	1 .10010001111000001010010
0: + 1: -	Decimal value of exponent field and exponent: 139 - 127 = 12	Decimal value of the significand: 1.5698340

Hexadecimal: 45C8F052 *Decimal:* 6430.0400

Example 3

Represent $3215.020002_{10}/4 = 803.7550005_{10}$

Decimal Value Entered: 803.7550005

Single precision (32 bits):

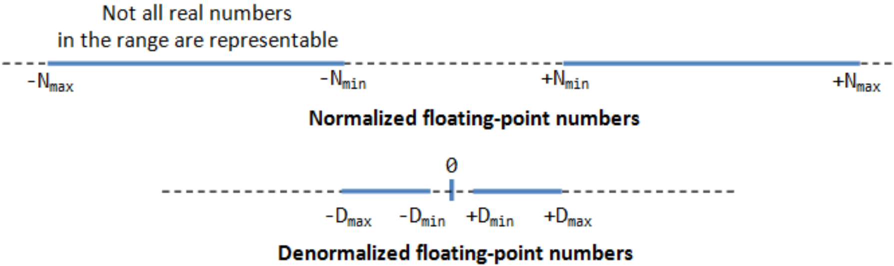
Binary: *Status:* normal

Bit 31 Sign Bit	Bits 30 - 23 Exponent Field	Bits 22 - 0 Significand
0	10001000	1 .10010001111000001010010
0: + 1: -	Decimal value of exponent field and exponent 136 - 127 = 9	Decimal value of the significand 1.5698340

Hexadecimal: 4448F052 *Decimal:* 803.75500

Floating-point numbers are auto-scaled!

De-normalized Form



- Normalized form has a serious problem.
- With an implicit leading 1 for the fraction, it cannot represent the number zero!
- De-normalized form is devised to represent zero and small numbers.
- $E = 0 \Rightarrow 0.F$, implicit leading 0.

Example

Represent $-3.4\text{E-}39_{10}$

Decimal Value Entered: -3.4e-39

Single precision (32 bits):

Binary: *Status:* denormalized

Bit 31 Sign Bit	Bits 30 - 23 Exponent Field	Bits 22 - 0 Significand
1	00000000	0.10010100000101111010001
0: + 1: -	Decimal value of exponent field and exponent	Decimal value of the significand
	0 - 127 = -127	0.5784800

Hexadecimal: 802505D1 *Decimal:* -3.3999999e-39

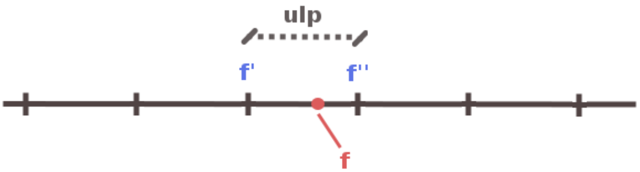
Special values

- **Zero**: $E = 0$, $F = 0$. Two representations: **+0** ($S = 0$) and **-0** ($S = 1$).
- **Inf** (Infinity): $E = 0\text{xFF}$, $F = 0$. Two representations: **+Inf** ($S = 0$) and **-Inf** ($S = 1$).
- **NaN** (Not a Number): $E = 0\text{xFF}$, $F \neq 0$. A value that cannot be represented as a real number (e.g. 0/0).

MATLAB

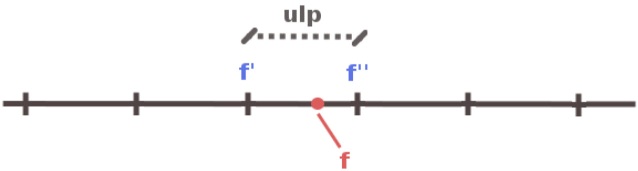
1. » a = 1/0
2. » ans = Inf
3. » b = −1/0
4. » ans = −Inf
5. » c = 0/0
6. » ans = NaN

Rounding schemes



- *ulp* (unit of least precision, `eps()`).
- f, significant, $f = 1.F$.
- f' and f'' being two successive multiples of ulp.
- Assume that $f' < f < f''$, $\; f'' = f' + ulp$,
- Then, the rounding function $round(f)$ associates to f either f' or f'', according to some rounding strategy.

Rounding schemes II



Rounding schemes are:

- *Truncation* (also called *round toward 0* or *chopping*):
  - if f is positive, $round(f) = f'$.
  - if f is negative, $round(-f) = f''$.
- *Round toward plus infinity*: $round(f) = f''$.
- *Round toward minus infinity*: $round(f) = f'$.
- *Round to nearest* (default):
  - if $f < f' + ulp/2$, $round(f) = f'$.
  - if $f > f' + ulp/2$, $round(f) = f''$.

Dynamic range

$$DR_{dB} \approx 6.02 \cdot 2^{b_E}$$

where b_E is the number of bits of E.

For single precision (32-bits):

$$DR_{dB} \approx 6.02 \cdot 2^{8} \approx 1541\,\text{dB}$$

For fixed-point Q31 (32-bits):

$$DR_{dB} \approx 6.02 \cdot 31 \approx 186\,\text{dB}$$

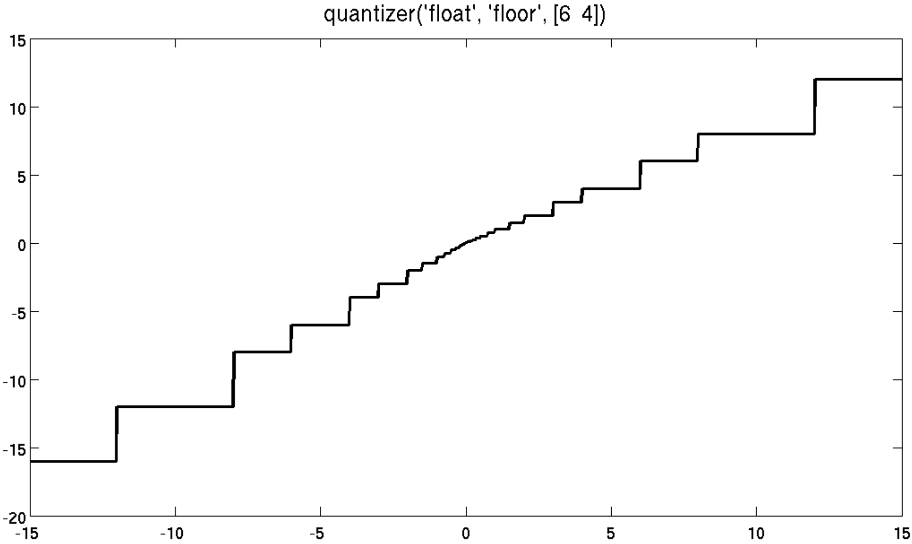
Precision

- Precision is not constant throughout floating point numbers' range.
- As the numbers get larger, the precision gets worse.

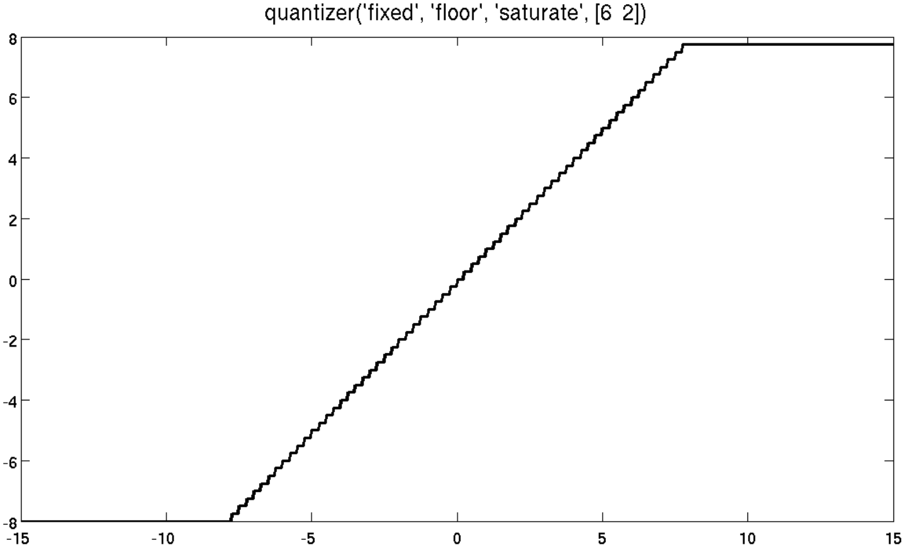
MATLAB

```
1 » u = linspace(-15,15,1000);
2 » q = quantizer([6 4],'float');  % [wordlength exponentlength]
3 » y1 = quantize(q,u);
4 » plot(u,y1); title(tostring(q))
5 »
6 » q = quantizer('fixed',[6 2]); % [wordlength fractionlength]
7 » y2 = quantize(q,u);
8 » plot(u,y2); title(tostring(q))
```

Precision II



Precision III



Precision IV

`eps(x)` returns the positive distance from `abs(x)` to the next larger in magnitude floating point number of the same precision.

MATLAB

1. » e1 = eps(single(1))
2. » e1 = 1.1920929e-07
3. » e2 = eps(single(1e1))
4. » e2 = 9.5367432e-07
5. » e3 = eps(single(1e10))
6. » e3 = 1024
7. » t = single(1e10) + single(1300)
8. » t = 10000001024.00

Sum of two floating-point positive numbers

$$n = n_1 + n_2 = 1.F \times r^{(E-bias)}\,,$$
$$n_1 = 1.F_1 \times r^{(E_1-bias)}\,,$$
$$n_2 = 1.F_2 \times r^{(E_2-bias)}\,.$$

- if $E_1 >= E_2$ then,

  $$E = E_1, \quad F = F_1 + (F_2 >> (E_1 - E_2))$$

- else,

  $$E = E_2, \quad F = (F_1 >> (E_2 - E_1)) + F_2$$

- if $F >= r$ then, (first normalization)

  $$E = E + 1, \quad F = F >> 1$$

- F = round(F)
- if $F >= r$ then, (second normalization)

  $$E = E + 1, \quad F = F >> 1$$

Example 1

$$n = 1e10 + 1300\,,$$

$$1e10 = (-1)^0 \times 1.00101010000001011111001 \times r^{(160-127)}\,,$$

$$1300 = (-1)^0 \times 1.11100000000000000000000 \times r^{(131-127)}\,.$$

- if $160 >= 131$ then,

$$E = 160,$$
$$F = 1.00101010000001011111001 + (1.11100000000000000000000 >> 29)$$

$$E = 160, \quad F = 1.00101010000001011111001 + (0.00000000000000000000000)$$

$$E = 160, \quad F = 1.00101010000001011111001$$

$$n = (-1)^0 \times 1.00101010000001011111001 \times r^{(160-127)}\,,$$

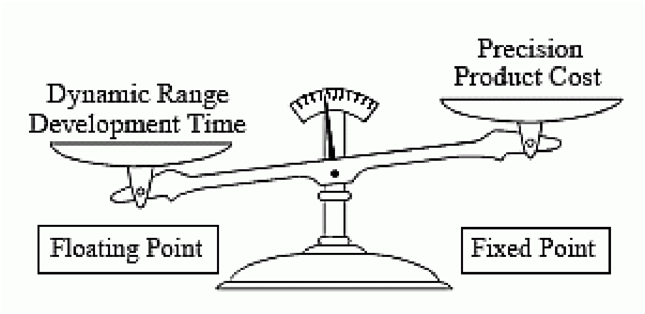
Example 2

- In floating-point processors scaling the data increases dynamic range, but scaling does not improve precision, and in fact degrades performance.
- When calculations involve large and small numbers at the same time, the loss of precision affects the small number and the result.

MATLAB

```
1 » (2^53 + 1) - 2^53
2 » ans = 0
3 » x=1, t = tan(x) - sin(x)/cos(x)
4 » t = 2.2204e-16 % eps(1)
```

Fixed-point vs floating-point



Bibliography

1. Bruno Paillard. An Introduction To Digital Signal Processors, Chapter 5 "Binary representations and fixed-point arithmetic".
2. Richard G. Lyons. Understanding Digital Signal, Chapter 12 "Digital data formats and their effects".
3. Texas Instruments. C28x IQ – Math Library. Link to document.
4. Jean-Pierre Deschamps, Gustavo D. Sutter, and Enrique Cantó. Guide to FPGA Implementation of Arithmetic Functions, Chapter 12 "Floating Point Arithmetic".