Errors

Scientific Notation

In **scientific notation**, a number can be expressed in the form

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

where r is a coefficient in the range $1 \le r < 10$ and m is the exponent.

$1165.7 =$ 1.1657×10^3

$0.0004728 =$

Error in Numerical Methods

- Every result we compute in Numerical Methods contain errors!
- We always have them… so our job? Reduce the impact of the errors
- How can we model the error?

x = true (exact) $\hat{x}$ = approx

$$e_a = |x - \hat{x}|$$

$$e_r = \frac{|x - \hat{x}|}{|x|}$$

- Absolute errors can be misleading, depending on the magnitude of the true value x.

- Example:

$\Delta x = 10^{-1}$

i) $x = 10^{5} \longrightarrow 10^{5} \pm 10^{-1}$ ✓

ii) $x = 10^{-5} \longrightarrow 10^{-5} \pm 10^{-1}$

- Relative error is independent of magnitude!

You are tasked with measuring the height of a tree which is known to be exactly 170 ft tall. You later realized that your measurement tools are somewhat faulty, up to a relative error of 10%. What is the maximum measurement for the tree height (numbers rounded to 3 sig figs)?

meet.ps/cs357

A) 153 ft

B) 155 ft

C) 187 ft

D) 189 ft

$x = 170$ ft

$e_r = 0.1$

$$e_r = \frac{|x - \hat{x}|}{|x|} \Rightarrow e_r|x| = |x - \hat{x}|$$

$$\hat{x} = x(1 \pm e_r)$$

$$= 170(1.1) = 187 \text{ ft}$$

You are tasked with measuring the height of a tree and you get the measurement as 170 ft tall. You later realized that your measurement tools are somewhat faulty, up to a relative error of 10%. What is the minimum height of the tree (numbers rounded to 3 sig figs) ?

A) 153 ft

B) 155 ft

C) 187 ft

D) 189 ft

$\hat{x} = 170 \text{ ft}$

$e_r = 0.1$

$\hat{x} = x(1 \pm e_r)$

$x = \dfrac{\hat{x}}{(1 \pm e_r)}$

$\dfrac{170}{1.1} = 154.54$

$\boxed{155 \text{ ft}}$

$\dfrac{170}{0.9} = 188.8$

$\boxed{189 \text{ ft}}$

Significant digits

Significant figures of a number are digits that carry meaningful information. They are digits beginning to the leftmost nonzero digit and ending with the rightmost "correct" digit, including final zeros that are exact.

The number 3.14159 has 6 significant digits.

The number 0.00035 has 2 significant digits.

The number 0.000350 has 3 significant digits.

Accurate to *n* significant digits means that you can trust a total of *n* digits. *Accurate digits* is a measure of relative error.

Suppose x is the true value and $\tilde{x}$ the approximation.

The number of significant digits tells us about how many positions of x and $\tilde{x}$ agree.

$\tilde{x}$ has $\boldsymbol{n}$ **significant figures** of x if $|x - \tilde{x}|$ has zeros in the first n decimal places counting from the leftmost nonzero (leading) digit of x, followed by a digit from 0 to 4.

Example:

$$x = 5.1 \; and \; \tilde{x} = 5$$
$$x = 0.51 \; and \; \tilde{x} = 0.5$$
$$x = 5 \; and \; \tilde{x} = 4.992$$
$$x = 5 \; and \; \tilde{x} = 4.996$$

$5.1 - 5 = 0.1 \rightarrow$ 1 sig fig

$0.51 - 0.5 = \boxed{0.01} \rightarrow$ 1 sig fig

$\boxed{5} - 4.992 = \underbrace{0.0}0 8 \rightarrow$ 2 sig fig

$5 - 4.996 = \underbrace{0.00}4 \rightarrow$ 3 sig fig

Suppose x is the true value and $\tilde{x}$ the approximation.

The number of significant digits tells us about how many positions of x and $\tilde{x}$ agree.

$\tilde{x}$ has $\boldsymbol{n}$ **significant figures** of x if $|x - \tilde{x}|$ has zeros in the first n decimal places counting from the leftmost nonzero (leading) digit of x, followed by a digit from 0 to 4.

Example:

$$x = 3.141592653$$

$\hat{x} = 3.14159 \times 10^{0}$ $\quad x - \hat{x} = \underbrace{0.000000}_{6 \text{ zeros}}2653$ $\quad 2.653 \times 10^{-6}$ → 6 sig figs

$\hat{x} = 3.1415$ $\quad x - \hat{x} = \underbrace{0.0000}_{4 \text{ zeros}}92653 \rightarrow 0.92653 \times 10^{-4}$ $\quad$ 5 sig figs

$\hat{x} = 3.1416$ $\quad x - \hat{x} = 0.000007347 = 0.7347 \times 10^{-5}$

$$\boxed{e_a = |x - \hat{x}| \leq 5 \times 10^{-n}}$$

Suppose x is the true value and $\tilde{x}$ the approximation.

What if instead we had $x = q \times 10^p$?

The number of significant digits tells us about how many positions of x and $\tilde{x}$ agree.

$\tilde{x}$ has **n significant figures** of x if $|x - \tilde{x}|$ has zeros in the first n decimal places counting from the leftmost nonzero (leading) digit of x, followed by a digit from 0 to 4.

Example:

$x = 3.141592653 \times 10^p$

$\hat{x} = 3.14159 \times 10^p \qquad x - \hat{x} = \underbrace{0.000000}_{6 \text{ zeros}}2653 \quad 2.653 \times 10^{-6} \times 10^p \quad \longrightarrow 6 \text{ sig figs}$

$\hat{x} = 3.1415 \qquad x - \hat{x} = \underbrace{0.0000}_{4 \text{ zeros}}92653 \longrightarrow 0.92653 \times 10^{-4} \times 10^p \quad 5 \text{ sig figs}$

$\hat{x} = 3.1416 \qquad x - \hat{x} = 0.000007347 = 0.7347 \times 10^{-5} \times 10^p$

$$\boxed{e_r = \frac{|x - \hat{x}|}{x} < \frac{5 \times 10^{-n} \times \cancel{10^p}}{q \times \cancel{10^p}}}$$

$1 \leq q < 10$

Accurate to *n* significant digits means that you can trust a total of *n* digits. *Accurate digits* is a measure of relative error.

Relative error: $$error = \frac{|x_{exact} - x_{approx}|}{|x_{exact}|} \leq 10^{-n+1}$$

n is the number of accurate significant digits

Hence if $rtol = 10^{-2}$ we need at least $n = 3$ sigfigs.

$$e_r \lesssim 5 \times 10^{-n}$$

$$e_r \lesssim 10 \times 10^{-n} \rightarrow \boxed{e_r \lesssim 10^{-n+1}}$$

After rounding, the resulting number has 5 accurate digits. What is the tightest estimate of the upper bound on my relative error?

$A)\ 10^5$

$B)\ 10^{-5}$

$e_r \lesssim 10^{-5+1} \Longrightarrow e_r \lesssim 10^{-4}$

$C)\ 10^4$

$D)\ 10^{-4}$

Sources of Error

Main source of errors in numerical computation:

- **Rounding error:** occurs when digits in a decimal point ($1/3 = 0.3333...$) are lost (0.3333) due to a limit on the memory available for storing one numerical value.

- **Truncation error:** occurs when discrete values are used to approximate a mathematical expression (eg. the approximation $\sin(\theta) \approx \theta$ for small angles θ)

Floating point representation

0.125 31.875

(Unsigned) Fixed-point representation

The numbers are stored with a fixed number of bits for the integer part and a fixed number of bits for the fractional part.

Suppose we have 8 bits to store a real number, where 5 bits store the integer part and 3 bits store the fractional part:

$$(\overbrace{1\;0\;1\;1\;1}^{\text{5 bits}}.\overbrace{0\;1\;1}^{\text{3 bits}})_2$$

$$2^4\;2^3\;2^2\;2^1\;2^0\;\;2^{-1}\;2^{-2}\;2^{-3}$$

2^{-1} 2^{-2} 2^{-3}

Smallest number: $(00000.001)_2 = 2^{-3} = (0.125)_{10}$

Largest number: $(11111.111)_2 = (31.875)_{10}$

(Unsigned) Fixed-point representation

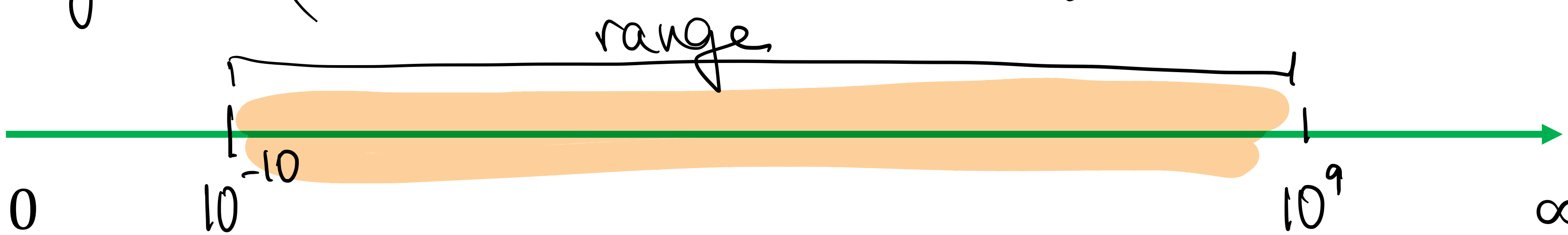
Suppose we have 64 bits to store a real number, where 32 bits store the integer part and 32 bits store the fractional part:

$$(a_{31} \dots a_2 a_1 a_0 . b_1 b_2 b_3 \dots b_{32})_2 = \sum_{k=0}^{31} a_k \, 2^k + \sum_{k=1}^{32} b_k \, 2^{-k}$$

$$= a_{31} \times 2^{31} + a_{30} \times 2^{30} + \cdots + a_0 \times 2^0 + b_1 \times 2^{-1} + b_2 \times 2^2 + \cdots + b_{32} \times 2^{-32}$$

smallest: $\underbrace{000 \dots 00}.\underbrace{\overset{2^{-1}}{0}\overset{2^{-2}}{0}\overset{2^{-3}}{0} \dots 00\overset{2^{-32}}{1}}_{32 \text{ bits}} = 2^{-32} \approx 10^{-10}$

largest: $(111 \dots 11.111 \dots 11) \approx 10^{9}$



(Unsigned) Fixed-point representation

Range: difference between the largest and smallest numbers possible.

More bits for the integer part $\longrightarrow$ increase range

Precision: smallest possible difference between any two numbers

More bits for the fractional part $\longrightarrow$ increase precision

$(a_2 a_1 a_0 . b_1 b_2 b_3)_2$ OR $(a_1 a_0 . b_1 b_2 b_3 b_4)_2$

Wherever we put the binary point, there is a trade-off between the amount of range and precision. **It can be hard to decide how much you need of each!**

Floating-point numbers

A floating-point number can represent numbers of different order of magnitude (very large and very small) with the same number of fixed bits.

In general, in the binary system, a floating number can be expressed as

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

q is the significand, normally a fractional value in the range $[1.0, 2.0)$

m is the exponent

Floating-point numbers

leading bit

fractional

n+1

Numerical Form:

$$x = \pm q \times 2^m = \pm b_0.\underbrace{b_1 b_2 b_3 \ldots b_n}_{\substack{\text{Fractional part of significand} \\ (n \text{ bits})}} \times 2^m$$

$b_i \in \{0,1\}$

Exponent range: $m \in [L, U]$

Precision: $\boxed{p = n + 1}$

total # of bits in significand

“Floating” the binary point

$$(1011.1)_2 = 1\times 8 + 0\times 4 + 1\times 2 + 1\times 1 + 1\times\frac{1}{2} = (11.5)_{10}$$

$$(10111)_2 = 1\times 16 + 0\times 8 + 1\times 4 + 1\times 2 + 1\times 1 = (23)_{10}$$

$$= (1011.1)_2\times 2^1 = (23)_{10}$$

$$(101.11)_2 = 1\times 4 + 0\times 2 + 1\times 1 + 1\times\frac{1}{2} + 1\times\frac{1}{4} = (5.75)_{10}$$

$$= (1011.1)_2\times 2^{-1} = (5.75)_{10}$$

Move “binary point” to the left by one bit position: Divide the decimal number by 2

Move “binary point” to the right by one bit position: Multiply the decimal number by 2

Converting floating points

Convert $(39.6875)_{10} = (100111.1011)_2$ into floating point representation

$$\boxed{1.001111011 \times 2^5}$$

fixed as one

Normalized floating-point numbers

Normalized floating point numbers are expressed as

$$x = \pm 1. b_1 b_2 b_3 \ldots b_n \times 2^m = \pm 1. f \times 2^m$$

where f is the fractional part of the significand, m is the exponent and $b_i \in \{0,1\}$.

- Fix the leading bit to 1
- range $m \in [L, U]$ (L: lower, U: upper)
- $\boxed{\text{precision} = n+1}$

hidden bit representation gains 1 bit!

\# bits "stored" = n

n is the number of bits in the fractional part

Iclicker question

Determine the normalized floating point representation $1.\boldsymbol{f} \times 2^{\boldsymbol{m}}$ of the decimal number $x = 47.125$ ($\boldsymbol{f}$ in binary representation and $\boldsymbol{m}$ in decimal)

A) $(1.01110001)_2 \times 2^{\mathbf{5}}$
B) $(1.01110001)_2 \times 2^{\mathbf{4}}$
C) $(1.01111001)_2 \times 2^{\mathbf{5}}$
D) $(1.01111001)_2 \times 2^{\mathbf{4}}$

Normalized floating-point numbers

$$x = \pm q \times 2^m = \pm 1.b_1 b_2 b_3 \ldots b_n \times 2^m = \pm 1.f \times 2^m$$

n: # bits fractional

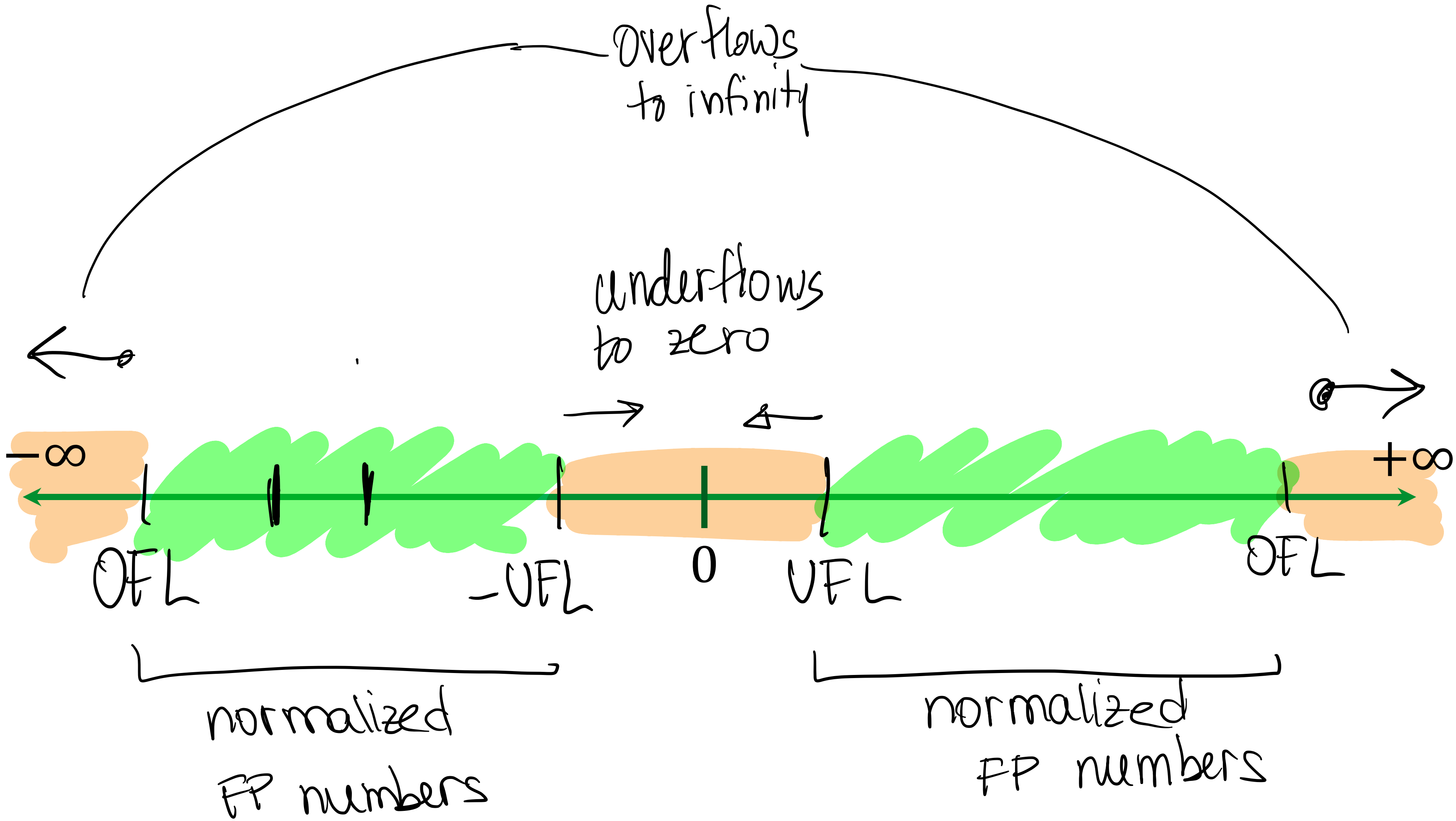
- **Exponent range**: $m \in [L, U]$
- **Precision**: $p = n+1$
- **Smallest positive normalized FP number:**

$x = 1.\underbrace{000 \cdots 0}_{n} \times 2^L$ $\boxed{UFL = 2^L}$

- **Largest positive normalized FP number:**

$x = 1.\underbrace{111 \cdots 11}_{n} \times 2^U \rightarrow$ $\boxed{OFL = 2^{U+1}(1-2^{-p})}$

Normalized floating point number scale



Floating-point numbers: Simple example

A "toy" number system can be represented as $x = \pm 1.b_1 b_2 \times 2^m$
for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

$m = 0$	$m = 1$	$m = 2$	$m = 3$	$m = 4$
$x = 1.00 \times 2^0$	1.00×2^1			
1.01×2^0				
1.10×2^0				
1.11×2^0	1.11×2^1			

$m = -1 \quad -2 \quad -3 \quad -4$

Floating-point numbers: Simple example

A "toy" number system can be represented as $x = \pm 1.b_1 b_2 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

$(1.00)_2 \times 2^0 = 1$
$(1.01)_2 \times 2^0 = 1.25$
$(1.10)_2 \times 2^0 = 1.5$
$(1.11)_2 \times 2^0 = 1.75$

$(1.00)_2 \times 2^1 = 2$
$(1.01)_2 \times 2^1 = 2.5$
$(1.10)_2 \times 2^1 = 3.0$
$(1.11)_2 \times 2^1 = 3.5$

$(1.00)_2 \times 2^2 = 4.0$
$(1.01)_2 \times 2^2 = 5.0$
$(1.10)_2 \times 2^2 = 6.0$
$(1.11)_2 \times 2^2 = 7.0$

$(1.00)_2 \times 2^3 = 8.0$
$(1.01)_2 \times 2^3 = 10.0$
$(1.10)_2 \times 2^3 = 12.0$
$(1.11)_2 \times 2^3 = 14.0$

$(1.00)_2 \times 2^4 = 16.0$
$(1.01)_2 \times 2^4 = 20.0$
$(1.10)_2 \times 2^4 = 24.0$
$\boxed{(1.11)_2 \times 2^4 = 28.0}$

$(1.00)_2 \times 2^{-1} = 0.5$
$(1.01)_2 \times 2^{-1} = 0.625$
$(1.10)_2 \times 2^{-1} = 0.75$
$(1.11)_2 \times 2^{-1} = 0.875$

$(1.00)_2 \times 2^{-2} = 0.25$
$(1.01)_2 \times 2^{-2} = 0.3125$
$(1.10)_2 \times 2^{-2} = 0.375$
$(1.11)_2 \times 2^{-2} = 0.4375$

$(1.00)_2 \times 2^{-3} = 0.125$
$(1.01)_2 \times 2^{-3} = 0.15625$
$(1.10)_2 \times 2^{-3} = 0.1875$
$(1.11)_2 \times 2^{-3} = 0.21875$

$(1.00)_2 \times 2^{-4} = 0.0625$
$(1.01)_2 \times 2^{-4} = 0.078125$
$(1.10)_2 \times 2^{-4} = 0.09375$
$(1.11)_2 \times 2^{-4} = 0.109375$

Same steps are performed to obtain the negative numbers. For simplicity, we will show only the positive numbers in this example.

$x = \pm 1.b_1 b_2 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$

not uniform distribution!

- Smallest normalized positive number: 2^{-L}

- Largest normalized positive number:

$$\boxed{2^{U+1}(1-2^{-p})}$$

$m \in [-4,4]$

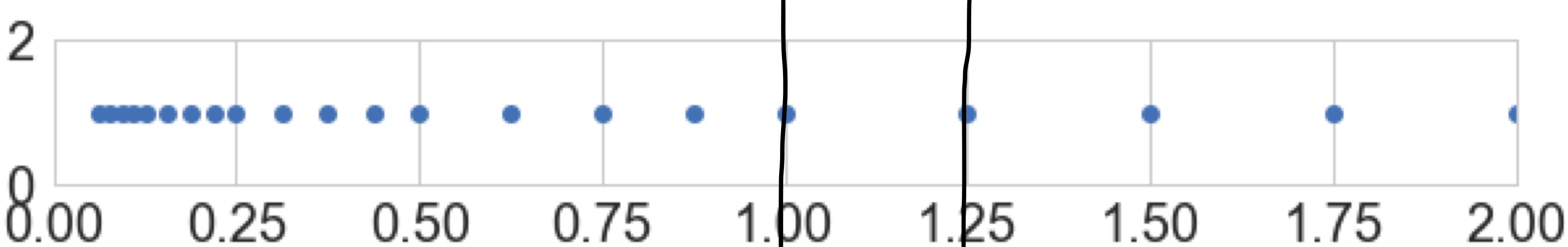
precision: $p = n+1 = 3$

Machine epsilon

$\boxed{\epsilon_m = 2^{-n}}$

- **Machine epsilon** (ϵ_m): is defined as the distance (gap) between 1 and the next largest floating point number.

$$x = \pm 1.b_1 b_2 \times 2^m \quad \text{for } m \in [-4,4] \text{ and } b_i \in \{0,1\}$$



$\epsilon_m = 0.25$

one : $1.\underbrace{000 \ldots 000}_{n \text{ bits}} \times 2^0$

: $1.\underbrace{000 \ldots 001}_{n\text{bits}} \times 2^0$

$0.\underbrace{000 \ldots 001}_{n \text{ bits}} \times 2^0$

$2^{-n} \times 2^0$

Machine numbers: how floating point numbers are stored?

Floating-point number representation

What do we need to store when representing floating point numbers in a computer?

$$x = \pm 1.f \times 2^{m}$$

$$x = \underbrace{\boxed{\;\pm\;}}_{\text{sign}}\underbrace{\boxed{\qquad m \qquad}}_{\text{exponent}}\underbrace{\boxed{\qquad\qquad f \qquad\qquad}}_{\text{significand}}$$

Initially, different floating-point representations were used in computers, generating inconsistent program behavior across different machines.

Around 1980s, computer manufacturers started adopting a standard representation for floating-point number: IEEE (Institute of Electrical and Electronics Engineers) 754 Standard.

Floating-point number representation

Numerical form:

$m \in [L, U]$

signed

$$x = \pm 1.f \times 2^{m}$$

Representation in memory:

$x =$

±	c	f

$c = m + \text{shift}$

unsigned

signed

STORES THE SHIFTED EXPONENT "c" (unsigned), INSTEAD OF ACTUAL EXPONENT "m"

Precisions:

IEEE-754 Single precision (32 bits):

		c	f
$x =$		8 bits	23 bits
	1bit		$b_1 b_2 b_3 \ldots b_{23}$

IEEE-754 Double precision (64 bits):

		c	f
$x =$	$\pm$	11 bits	52 bits
	1bit		$b_1 b_2 b_3 \ldots b_{52}$

Single $\Rightarrow$ 8 bits to store "C"

$(00000000)_2 = 0$

$(11111111)_2 = 255$

$\hookrightarrow$ Special cases (save for later!)

$1 \leq m+shift \leq 254$

$\Rightarrow shift = 127$

$\boxed{-126 \leq m \leq 127}$ $m \in [L, U]$