Announcements

- Questions re/labs → TAs
- Yes, lab02 is due THIS week
- In case of difficulty finishing assignments → talk to TA first
- Don't worry about the diary grades. They may show up late, or as a zero if the TA bundles grades.
- First homework → next week

Numbers

- Significant digits
- Accuracy and precision
- Number systems
- Quantization error

A simple example: add 0.1 repeatedly 100,000 times

We know the answer to this:

$$\sum_{k=1}^{100,000} 0.1 = 10,000$$

This is the answer my computer gave when I used about a number scheme that had about 7 decimal digits of precision:

$$\sum_{k=1}^{100,000} 0.1 = 9,998.556640625$$

This is the answer my computer gave when I used about a number scheme that had about 16 decimal digits of precision*:

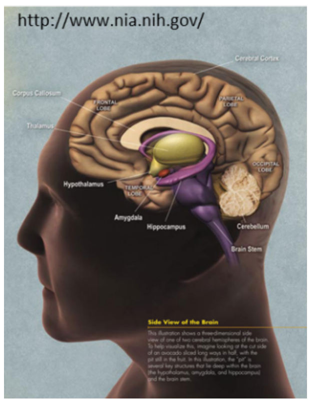
$$\sum_{k=1}^{100,000} 0.1 = 10,000.000000188480000$$

*This is the MATLAB default.

10,000.000000188480000 ≠ 10,000

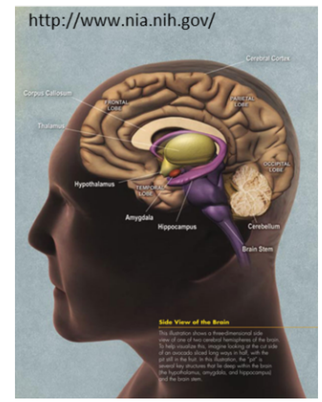
So what happened?

We do arithmetic using decimal numbers, so this is how we also tend to define our instructions to the computer.



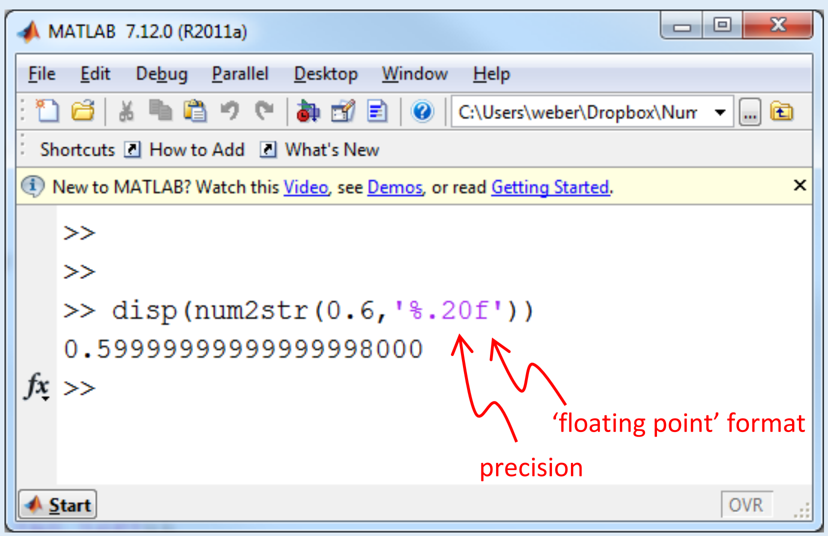
Almost all computers use binary numbers
...0110 0011 1000...

We also prefer to get our answers in the number system we are used to (decimal)

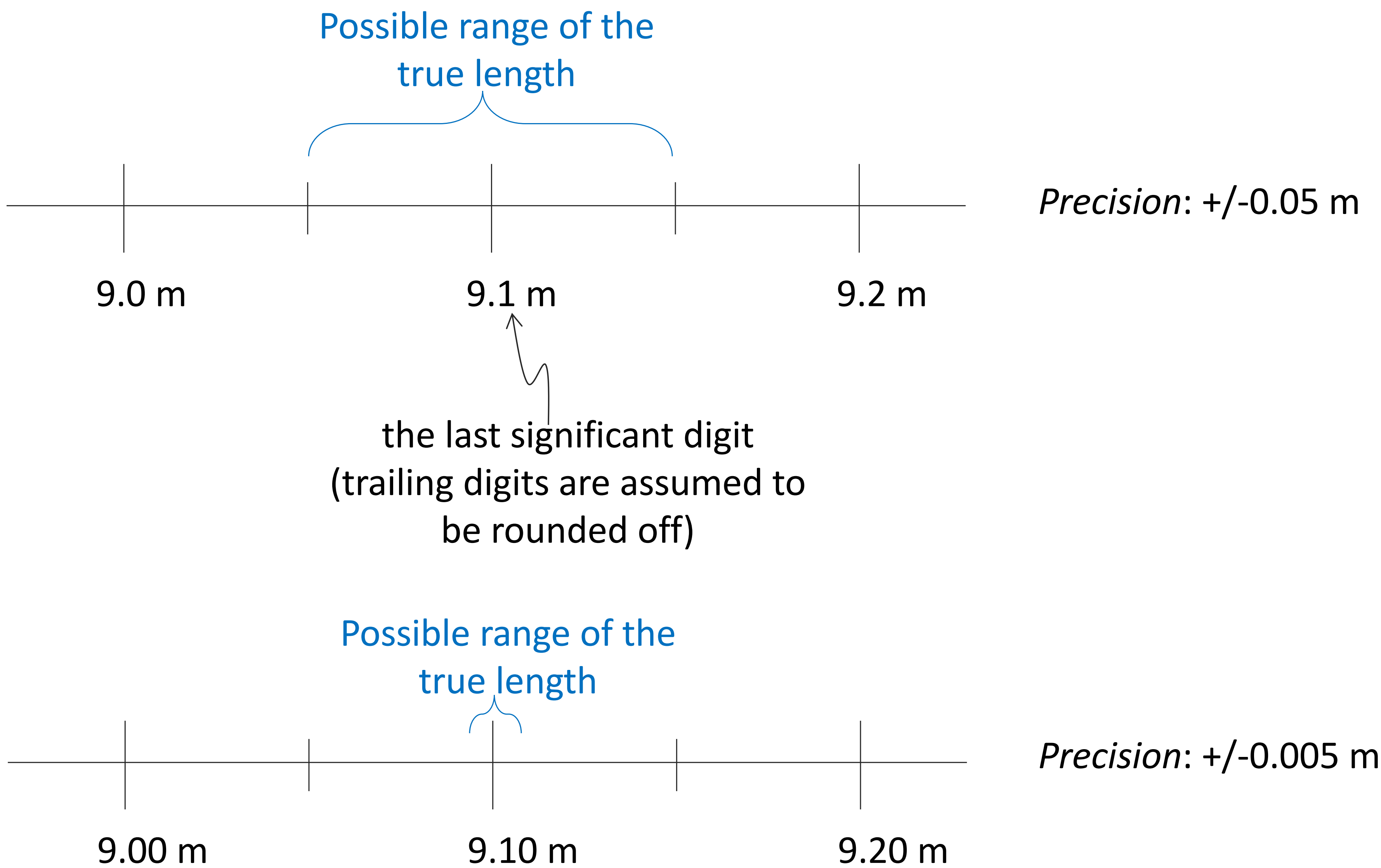


Some things are lost in translation.

0.1	0.10000000000000001000
0.2	0.20000000000000001000
0.3	0.30000000000000004000
0.4	0.40000000000000002000
0.5	0.50000000000000000000
0.6	0.59999999999999998000
0.7	0.70000000000000007000
0.8	0.80000000000000004000
0.9	0.90000000000000002000
1.0	1.00000000000000000000

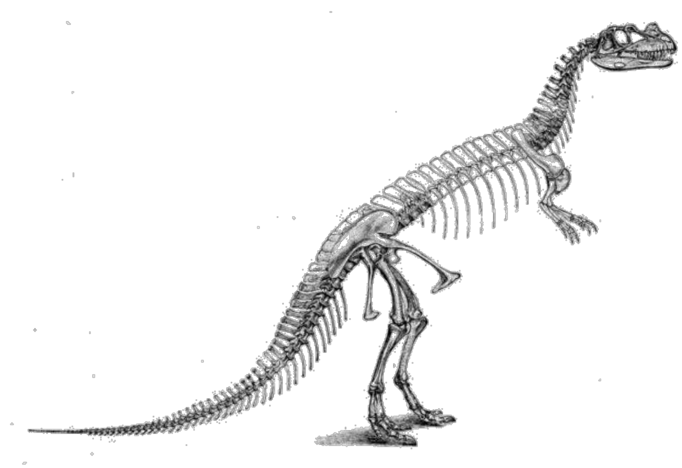


A significant digit: one that is known to be correct and reliable



These all have three significant digits:

0.716
.000716
716
7.16e5



These have four significant digits:

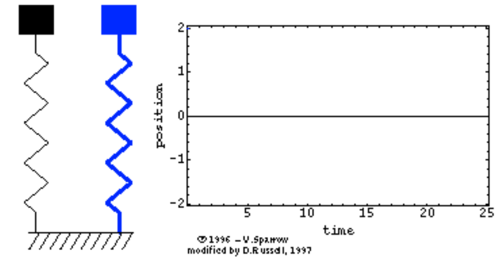
7.160e5
0.7160

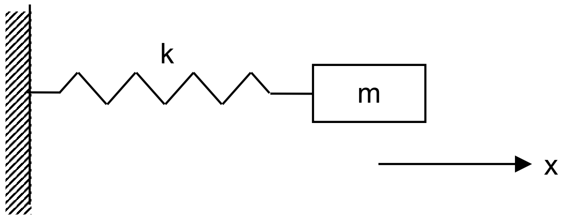
This one is tricky (ambiguous):

716000

Has it been rounded?
Is it precise to the nearest 1000?
Is it precise to the nearest 1?

Calculations with significant digits





$m = 1.1$ kg, $k = 4350.3142$ N/m

$$f_o = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \quad = 10.00884817594446 1\ \text{Hz}$$

Does this seem reasonable? The least *precise* data is given with two significant digits, so our answer should be also be given with two significant digits: $f_o = 10$ Hz

A general rule:
Stated results should typically be of the same order of magnitude as the uncertainty. That is, we don't use more significant digits than we're sure about.

Important Note:
When doing calculations on the way to an answer, you should normally be using at least one extra significant digit, and rounded at the end for the final answer.

Accuracy and Precision
(error analysis)

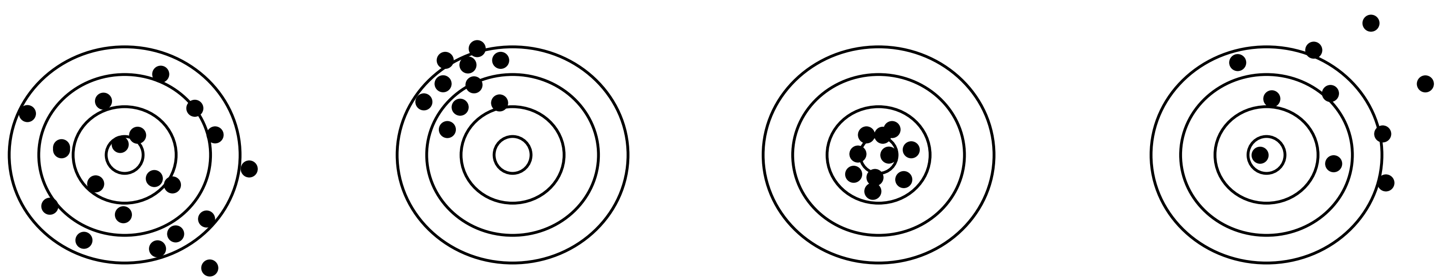
Blunders or mistakes:
- Transposed numbers
- wrong units
- incorrect decimal places

A blunder:
Incorrect unit conversion: needed 22,300 kg of fuel and got 22,300 pounds (10,100 kg)

Air Canada Flight 143:
"Gimli Glider"

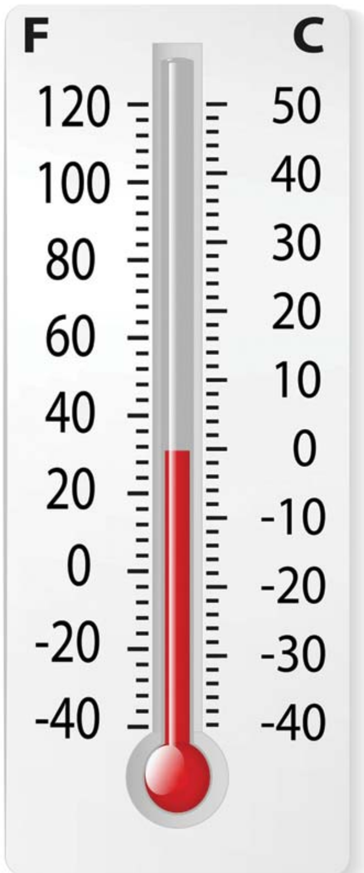
Discrepancies or disagreements
- The world is flat. No, its round.
  (actually, its an oblate spheroid)

Uncertainty
- systematic error (bias)
- random errors



Accurate Not precise | Inaccurate Precise | Accurate Precise | Inaccurate Not precise

Absolute and relative error



Truth: $T_t = 1°C$

Measurement: $T_a = 0°C$

True error: $\epsilon = |T_t - T_a|$ 1°

Relative error: $\eta = \dfrac{|T_t - T_a|}{|T_t|} = \dfrac{\epsilon}{|T_t|}$ 1

Percent error: $\dfrac{\epsilon}{|T_t|} \times 100$

Problem: Do we ever know the truth?

Questions for further thought:

What is the relative error in the example above if the truth is 10°? Or 100 °?

What is the relative error if the truth is 0°?

An example (where we don't know the truth)

Binomial expansion

$$(1+x)^a = 1 + ax + \frac{a(a-1)}{2!}x^2 + \frac{a(a-1)(a-2)}{3!}x^3 + \frac{a(a-1)(a-2)(a-3)}{4!}x^3 + \cdots$$

$$|x| < 1, a \neq 0, real$$

Compute $1.4^{3.1}$ to within 10% approximate error, using as few terms as possible

First approx.: $(1+0.4)^{3.1} = 1 + 3.1 * 0.4$ = 2.24

Second approx.: $(1+0.4)^{3.1} = 1 + 3.1 * 0.4 + \frac{3.1(3.1-1)}{2!}0.4^2$ = 2.76

|2.24-2.76|/2.76 = 0.19 > 10%

Third approx.: $(1+0.4)^{3.1} = 1 + 3.1 * 0.4 + \frac{3.1(3.1-1)}{2!}0.4^2 + \frac{3.1(3.1-1)(3.1-2)}{3!}0.4^3$ = 2.84

|2.76-2.84|/2.84 = 0.028 < 10% ✓

Numbers in Computers

Base-10:

247 → $2*10^2 + 4*10 + 7*10^0$

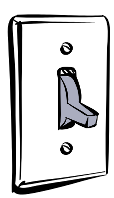
Base-2:

247 → $1*2^7 + 1*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$

(128 + 64 + 32 + 16 + 0 + 4 + 2 + 1)

11110111

These are bits. 8 bits = 1 byte.



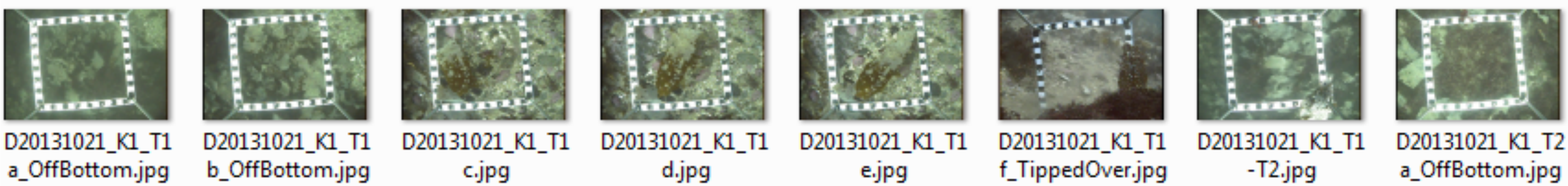
Lots of switches
(transistor)

Numbers in Computers

	signed	unsigned
8-bit (1-byte) binary numbers can represent these integers:	-128 → 127 C: char; Fortran: INTEGER*1	0→255
16-bit (2-byte) binary numbers can represent these integers:	-32,768 → 32,767 C: short int Fortran: INTEGER*2	0→65,537
24-bit (3-byte) binary numbers can represent these integers:	-8,388,608 → 8,388,607 Not used	0→ 16,777,215
32-bit (4-byte) binary numbers can represent these integers:	~+-10^9 C: int Fortran: INTEGER, INTEGER*4	0→ 4,294,967,295 C: unsigned int
64-bit (8-byte) binary numbers can represent these integers:	~+-10^{18} C: long int Fortran: INTEGER*8	0→ 1.84467440737096e+19 C: unsigned long int

Efficiency of Binary Numbers

1 2 3 4 5 6 7 8



To access these photos, I need 8 unique numbers using base-10.

But, what if we address these photos using a binary representation of their address?

The address for picture 5:

X		X
+4	+2	+1

3 unique 'slots'

That's not bad for 8 locations. What is the efficiency for 64,000 locations?

Numbers in Computers

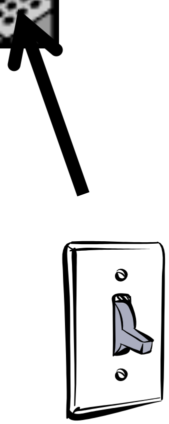
Base-10:

247.13 → $2*10^2 + 4*10 + 7*10^0 + 1*10^{-1} + 3*10^{-2}$

Base-2:

247 → 1111 0111

247.13 → ???



Lots of switches (transistor)

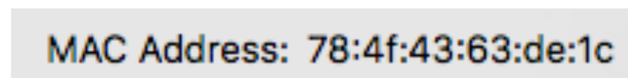
Octal/Hexadecimal/Base 256

Base-8 (octal):

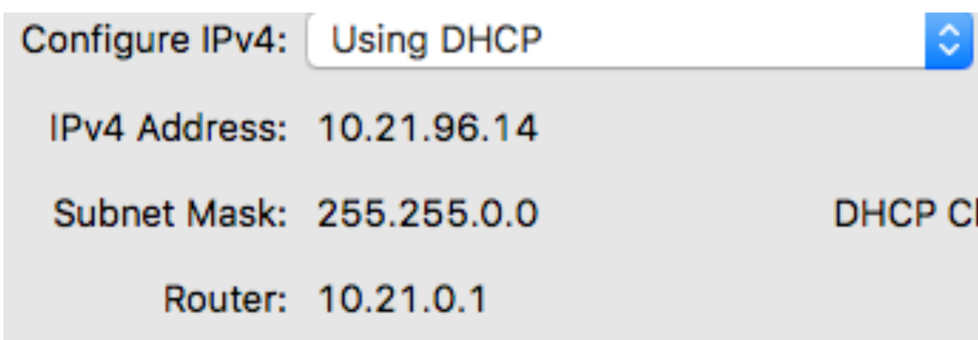
Group 3 bits in binary: 110 010 101 010 → o6252 or 06252
No digit larger than 7!

Base-16 (hexadecimal):

Group 4 bits in binary: 1100 1010 0010 → 0xDA2
The digits are now: 0123456789ABCDEF



Base-256 (IP addresses):



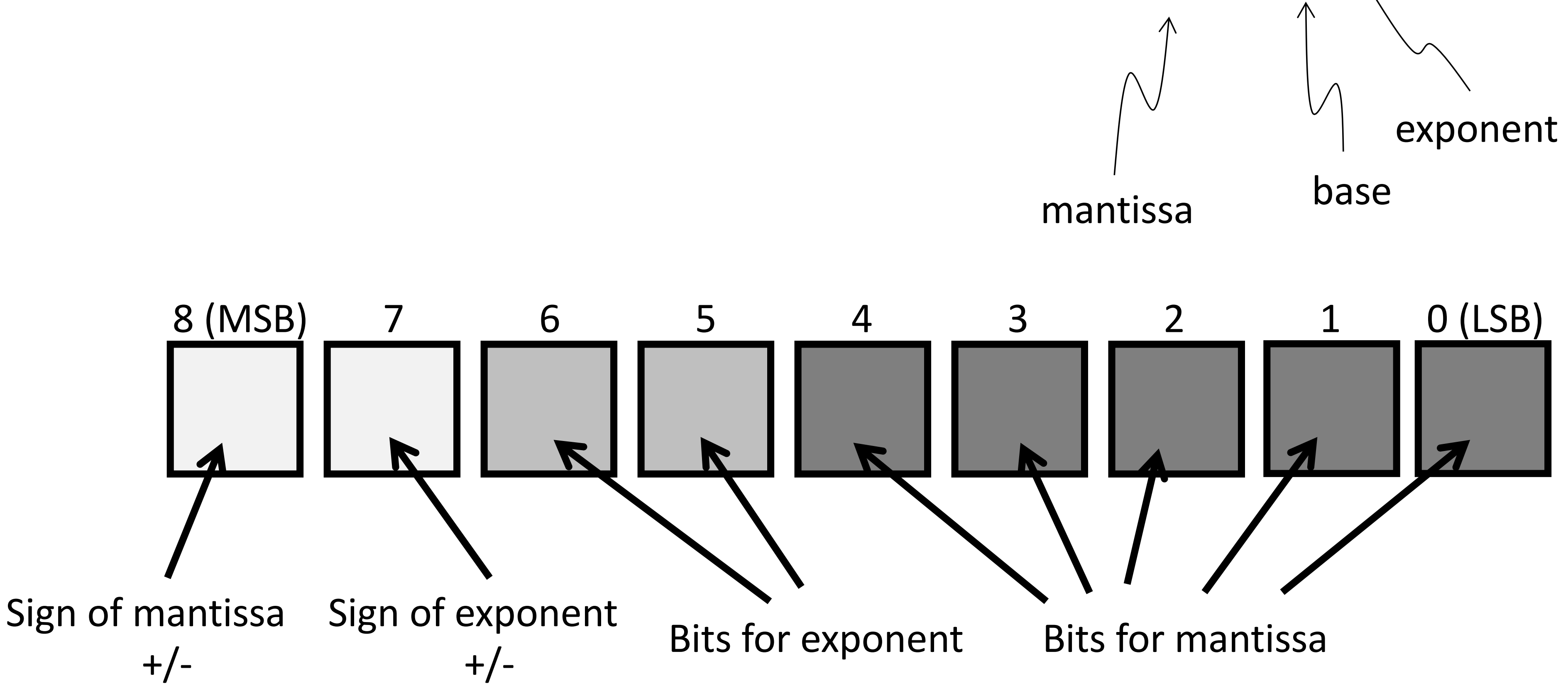
→ Integer numbers on a computer have a limited range!

→ Integer numbers have a constant true error (0.5)!

What can we do if we need larger numbers?

Floating Point Numbers

A floating point number: $3.928 = 3928 \times 10^{-3}$

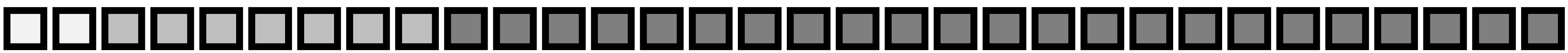


Questions:
Can I represent 3.928 with the ‘boxes’ (or bits) shown above?
What is the precision of this floating point number?

Anatomy of a 32-bit (a.k.a. single) Floating Point Number

$3.928 = 3928 \times 10^{-3}$

mantissa, base, exponent



2 sign bits

7 bits for representing the exponent

23 bits for representing the mantissa

Range of exponents: 0 → 2^7-1

Range of mantissa: 0 → 2^{23}-1

10^{-39} → 10^{38}, ~ 7 digits of precision

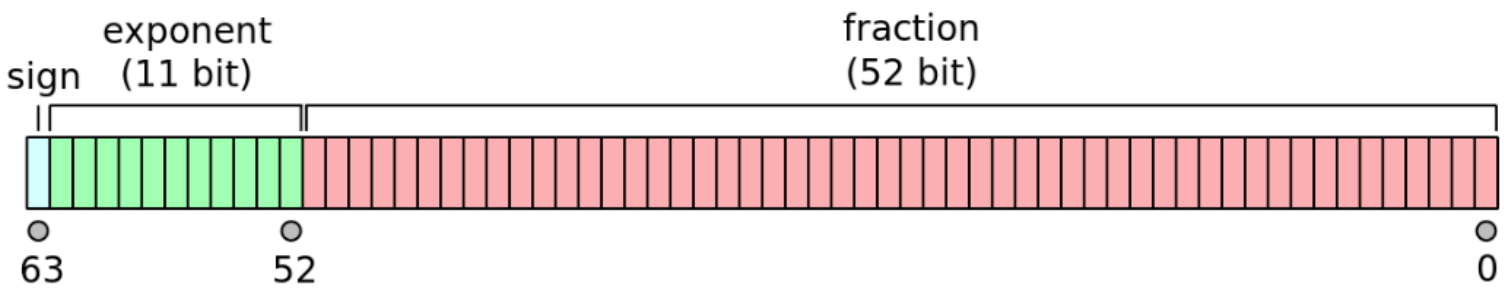
64-bit ('double') floating point number:

11-bit exponent
52-bit mantissa
2 sign

10^{-308} → 10^{308}, ~ 16 digits of precision

64 bit (8-byte) floating point number

IEEE 754 standard, used internally by MATLAB



The real value assumed by a given 64-bit double-precision datum with a given biased exponent e and a 52-bit fraction is

$$(-1)^{\text{sign}}(1.b_{51}b_{50}\ldots b_0)_2 \times 2^{e-1023}$$

or

$$(-1)^{\text{sign}}\left(1+\sum_{i=1}^{52} b_{52-i}2^{-i}\right) \times 2^{e-1023}$$

Between 2^{52}=4,503,599,627,370,496 and 2^{53}=9,007,199,254,740,992 the representable numbers are exactly the integers. For the next range, from 2^{53} to 2^{54}, everything is multiplied by 2, so the representable numbers are the even ones, etc. Conversely, for the previous range from 2^{51} to 2^{52}, the spacing is 0.5, etc.

MATLAB example: very small numbers (de-normalized numbers, gradual underflow

```
>> x = 10^(-307)

x =

   9.999999999999999e-308

>> x = 10^(-308)

x =

   9.999999999999999e-309

>> x = 10^(-309)

x =

   1.000000000000002e-309
```

Starting to loose precision

```
>> x = 10^(-314)

x =

   9.999999999638808e-315

>> x = 10^(-315)

x =

   9.999999984816838e-316

>> x = 10^(-316)

x =

   9.999999836597144e-317
```

Poor precision (not 16 digits!)

```
>> x = 10^(-322)

x =

   9.881312916824931e-323

>> x = 10^(-323)

x =

   9.881312916824931e-324

>> x = 10^(-324)

x =

     0
```

Very poor precision (not 16 digits!)

→ Floating point numbers have a much larger range than integers with the same storage requirement

→ Floating point numbers have a (more or less) constant relative error (precision)

→ Only a very limited subset of real numbers can be represented on a computer

Sick cases (usually coding error)

1/0, 2e222^2 → Floating point overflow → Inf

2e-222^2→ Floating point underflow → 0

0/0 → Makes no sense → NaN

MATLAB example

$$\frac{1.1 \times 10^{-exponent}}{1.0 \times 10^{-exponent}} = 1.1$$

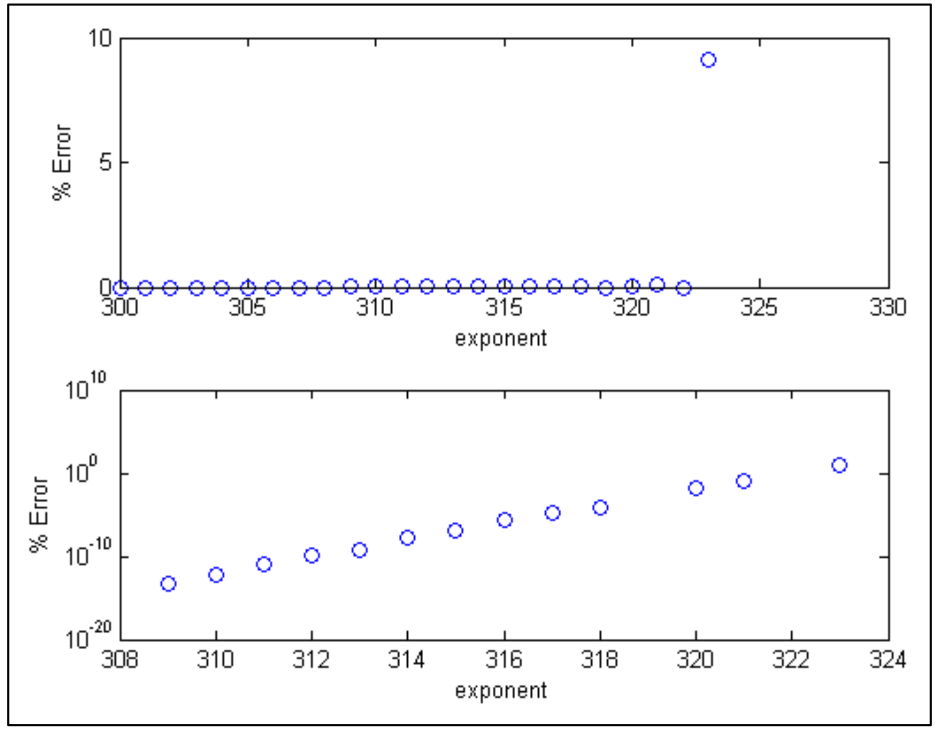
```
clear all; close all;

% this is a simple matlab script to examine the percent error
%  in very small numbers using MATLAB's native doubl precision
%  number scheme.
%
%  The script examines the difference in the ratio
%
%        1.1 x 10^-exponent
%       --------------------  = 1.10000000000000000
%        1.0 x 10^-exponent
%
exponent = 300:330;
for i = 1:length(exponent)
    num = 1.1*10^(-exponent(i));
    den = 1.0*10^(-exponent(i));
    x(i) = num/den;
end
perError = 100*abs(x-1.1)/1.1;      % this is the percent error

% plot the result in a linear plot (is this hard to see?)
subplot(211)
plot(exponent,perError,'o')
xlabel('exponent')
ylabel('% Error')

% plot the result in a log plot
subplot(212)
semilogy(exponent,perError,'o')
xlabel('exponent')
ylabel('% Error')
```

Results



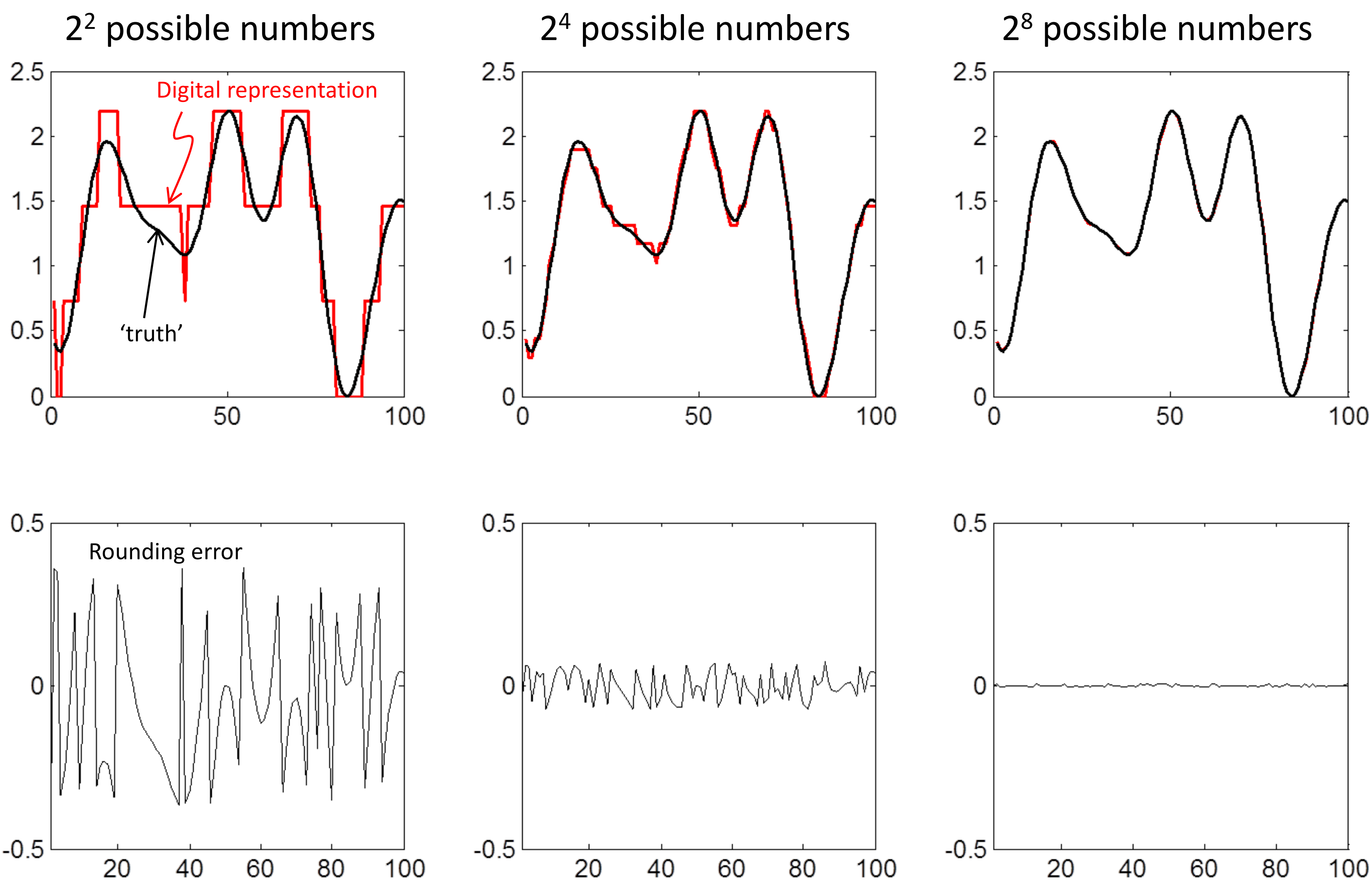
Question:

The MATLAB code uses exponents as high as 330 (10^-330). Why don't we see this in our plot? What does MATLAB give as a % error for these high exponents?

Quantization Error

Limitations in precision leads to truncation or **rounding**

Example: audio sampling (16 bit ADC typically)

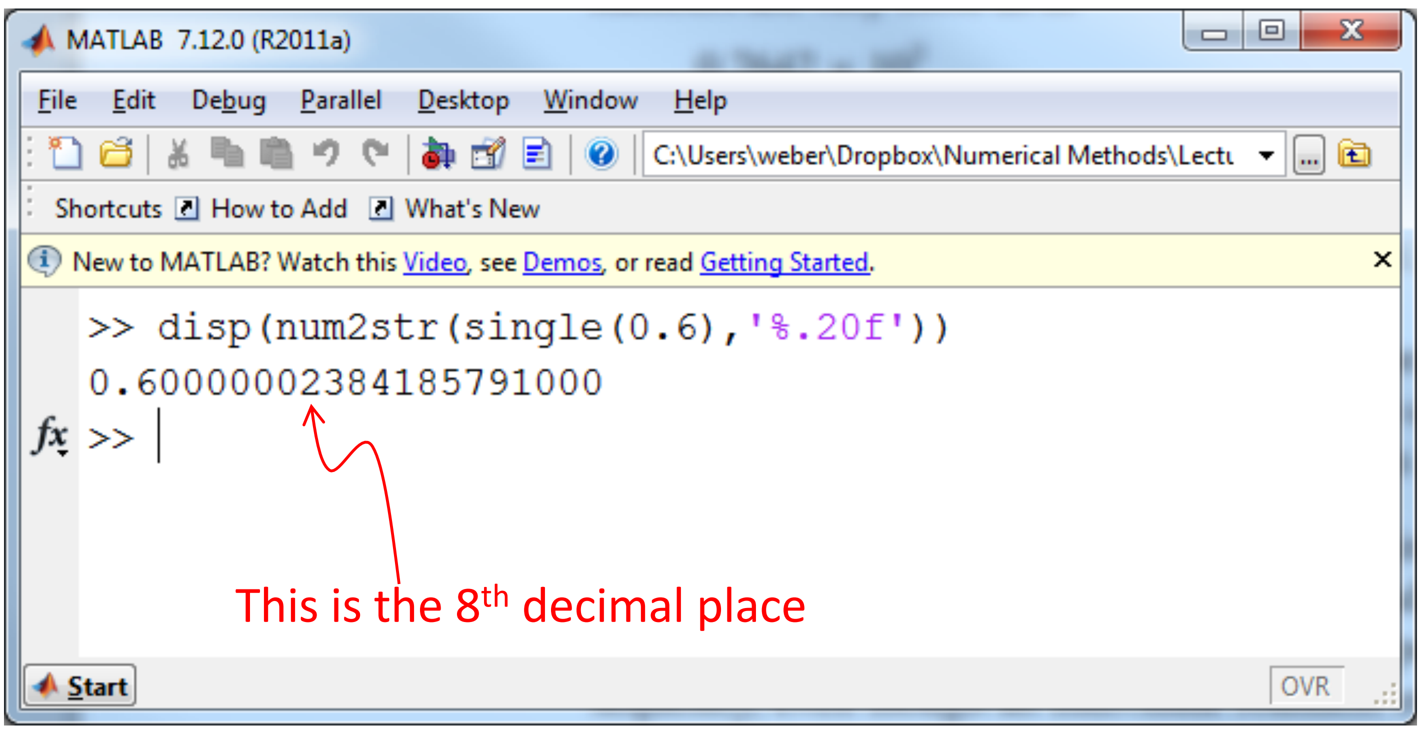


Return to the first example:

This is the answer my computer gave when I used about a number scheme that had about 7 decimal digits of precision:

$$\sum_{k=1}^{100{,}000} 0.1 = 9{,}998.556640625$$

The individual error in precision is small, but because were doing a large number of computations that depend on the results of earlier ones, the error grows large:



How to represent text?

Each character is one byte → ascii (American Standard Code for Information Interchange) table. 0-31 are control characters, 128-255 are extras, some are not printable.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

How to represent text and other stuff?

One line of text → ‘text’<lf> or ‘text<cr><lf>

End of file → <ctrl>Z (26) only Microsoft

How to store a picture → 3 bytes per pixel rgbrgbrgb..... Then compress (jpg)

How to store sound → 16bit (2byte) samples at 44,200 Hz → 5304000 bytes/minute → ~100 minutes on a CDROM (wav file).

Number Representations in Computers

character
'char'

A 1-byte individual character

ASCII Characters:
! " # $ % & ' ( ) * + , - . / 0 1
2 3 4 5 6 7 8 9 : ; < = > ? @ A
B C D E F G H I J K L M N O P
Q R S T U V W X Y Z [\ ] ^ _ `
a b c d e f g h I j k l m n o p q
r s t u v w x y z { | } ~

In the MATLAB command window:
>> char(32:126)

integer
int

An integer value

Either signed:
…-5,-4,-3,-2,-1,0,1,2,3,4,…

or unsigned:
0,1,2,3,4,5,…

The number of values that can be represented depends on the number of bytes:
1 byte (unsigned):
0→255
2 byte (short, unsigned):
0→65,535
4 byte (long, unsigned):
0→4,294,967,296

floating point
point

A floating point value

Single (4 bytes)
- 6-9 significant decimal points
- max value is ~3x10^{38}

Double (8 bytes)
- 15-17 significant decimal points
- max value is ~1x10^{308}

Quadruple (16 bytes)

Reference: Kernighan, B. W., and D. M. Ritchie, *The C Programming Language*, Second Edition, Prentice-Hall, Inc., 1988.

Unit Systems

http://physics.nist.gov/Pubs/SP330/sp330.pdf

Table 1. SI base units

Base quantity		SI base unit	
Name	Symbol	Name	Symbol
length	l, x, r, etc.	meter	m
mass	m	kilogram	kg
time, duration	t	second	s
electric current	I, i	ampere	A
thermodynamic temperature	T	kelvin	K
amount of substance	n	mole	mol
luminous intensity	I_v	candela	cd

Air Canada Flight 143: "Gimli Glider"

TABLE 1.1 SI Units

Quantity	Name of unit	Symbol	Equivalent
Length	Meter	m	
Mass	Kilogram	kg	
Time	Second	s	
Temperature	Kelvin	K	
Frequency	Hertz	Hz	s^{-1}
Force	Newton	N	$kg\ ms^{-2}$
Pressure	Pascal	Pa	$N\ m^{-2}$
Energy	Joule	J	N m
Power	Watt	W	$J\ s^{-1}$

TABLE 1.2 Common Prefixes

Prefix	Symbol	Multiple
Mega	M	10^6
Kilo	k	10^3
Deci	d	10^{-1}
Centi	c	10^{-2}
Milli	m	10^{-3}
Micro	μ	10^{-6}

From Kundu and Cohen, Fluid Mechanics.

Unit Systems

European scientists estimate that 100 kg of is wasted per person per year.

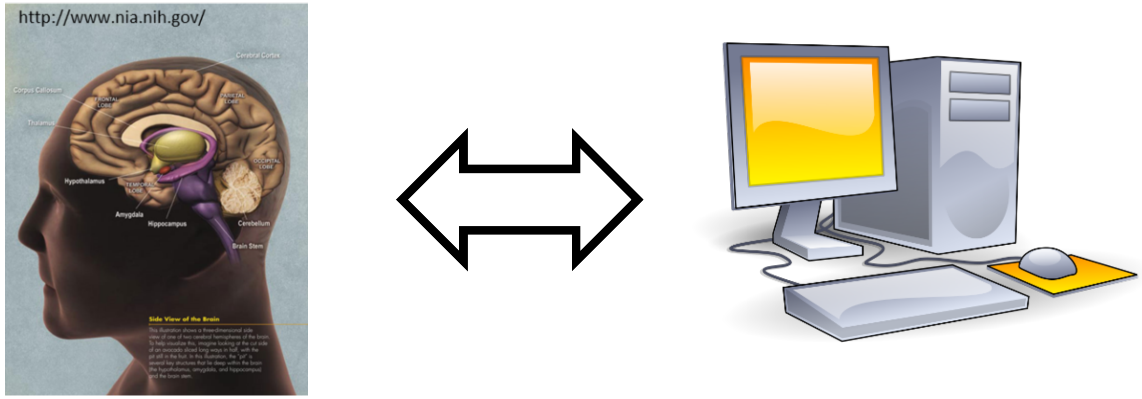
Unit conversion:
1 kg = 2.2 lbs

European scientists estimate that 220 lbs of food is wasted per person per year.

This is probably accurate to the nearest 50 kg or so, based on the available information in this sentence.

This is probably accurate to the nearest 5 lbs or so, based on the available information in this sentence.

A better way to say this: European scientists estimate that 100 kg (about 220 lbs) of food is wasted…



Take home message: pay attention to what you are asking your computer, and to what your computer is give you for an answer.

- Significant digits
- Accuracy and precision
- Number systems
- Quantization error