More technical matters: Floating-pt. arithmetic

What guarantees the "correct" solution of <u>any</u> numerical algorithm?

1. Bug-free code (obviously)
2. Convergent numerical algorithm (in infinite-precision)
3. "Stable" computations (recall $\cos(2\pi 10^{20})$?)

1 & 2 are "human" problems

3 is a "computer" problem

[Intel Flaw incorrectly computing quotients
$\Rightarrow$ \$420 million dollars

[Ariane 5 rocket - programmers didn't allocate enough memory]
$\Rightarrow$ all arrays at least 10 000 long!

The question you should be asking yourself is:
How do computers store numbers?

$\rightarrow$ Binary / base-2 arithmetic

Decimal: $10 = 1\cdot 10 + 0\cdot 1$

Binary: $1010 = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0$

Binary addition:

$$23 = 16 + 7$$
$$= 16 + 4 + 2 + 1$$
$$= 2^4 + 2^2 + 2^1 + 2^0$$
$$= (10111)_2$$

$$17 = 16 + 1$$
$$= 2^4 + 2^0$$
$$= (10001)_2$$

$$17 + 23 = 40 = 32 + 8$$
$$= 2^5 + 2^3$$
$$= (101000)_2$$

$$\begin{array}{r} \overset{1\,1\,1}{10111} \\ +\ 10001 \\ \hline \boxed{101000} \end{array}$$

Binary decimals:

$\frac{7}{2} = 3.5$ in decimal notation

In binary notation: $3 = (11)_2$

$$.5 = (0.1)_2$$
$$\frac{1}{2}$$
$$.25 = \frac{1}{4} = (0.01)_2$$
$$\frac{1}{2^2}$$

Each 0 or 1 is a "bit".

Fixed representation: XXXX.XXXX   8 bit fixed

Not how computers store numbers

Floating point representation (like scientific notation)

$$\pm m \times 2^E \quad , \quad m \in [1,2)$$

Example: $23 = (10111)_2$

$= |1.0111 \times 2^4|$ Every multiplication by 2 moves the decimal over by one.

$\frac{1}{8} = 0.001 = 1.0 \times 2^{-3}$

How many bits does a computer use?

single precision "word" : 32 bits
1 bit for sign
8 bits for exponent
23 bits for mantissa (significand)

Ex: $5.5 = 1.011 \times 2^2$

$= (\underset{\text{sign}}{0} \mid \underbrace{E = 00000010}_{\text{exponent}} \mid \underbrace{1011 \ldots 0}_{\text{Mantissa}})$

always 1 (can improve, except for 0)
hidden-bit rep.

Non-repeating ~~decimals~~ binary are "floating-point numbers"

What is the precision of this?

<u>Machine precision</u> is the distance between <u>1</u> and the next fp number. In this case,

$$1 = (0 \mid E=0 \mid 1.\overbrace{0\ldots\ldots0}^{23\text{ bits}})$$

$\frac{1}{2^{23}}$

$$1 + 2^{-23} = (0 \mid E=0 \mid 1.0\cdots01)$$

$\approx 1.2 \times 10^{-7}$