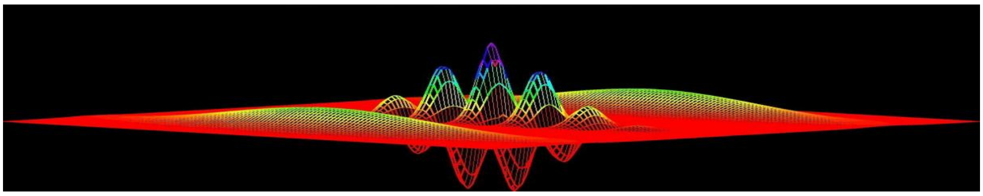
Computational Physics

numerical methods with C++ (and UNIX)



Fernando Barao
Instituto Superior Tecnico, Dep. Fisica
email: barao@lip.pt

computer storage precision

✔ the number of bytes assigned to a real variable (word length) is controlled by the programmer

single precision			
4 Bytes	s(1)	e(8)	m(23)
double precision			
8 Bytes	s(1)	e(11)	m(52)

accuracy

single

$2^{-23} \sim 10^{-8}$

double

$2^{-52} \sim 10^{-16}$

max/min values

single

$2^{127} \simeq 1.7 \times 10^{38}$

$2^{-127}\ 2^{-23} \sim \times 10^{-45}$

✔ **round-off errors**

- a real number with a finite number of digits in the decimal system can require an infinite number of bits in the binary system

$$6.28125 \quad = \quad 0\ 10000001\ 10010010000000000000000 \quad \text{(precise)}$$

$$0.42 \quad = \quad 0\ 01111101\ 10101110000101000111101 \quad \text{(round-off)}$$

Types of errors

- ✔ approximation errors

  errors resulting from the problem simplification in order to be solved on the computer

  continuous functions are approximated by finite arrays of values

  - ▶ problem discretization
  - ▶ replacement of a infinite series by a sum for finite terms

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \simeq \sum_{n=0}^{N} \frac{x^n}{n!} = e^x + \Delta(x, N)$$

- ✔ round-off errors

  errors arising from using a finite number of digits to represent real numbers

Example : derivative computation

- ✔ **Function Taylor expansion :**

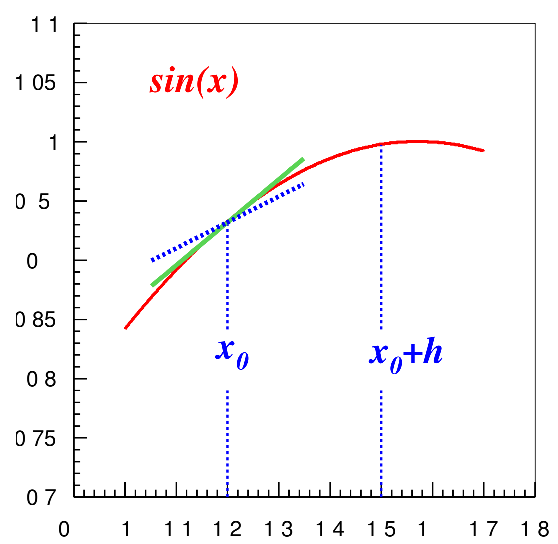
$$f(x_0 + h) = f(x_0) + hf^{'}(x_0) + \frac{h^2}{2} f''(x_0) + \cdots + \frac{h^k}{k!} f^{(k)}(x_0) + \cdots$$

- ✔ **The derivative of the function can be calculated as :**

$$f^{'}(x_0) \simeq \frac{f(x_0 + h) - f(x_0)}{h}$$

- ✔ **The discretization error :**

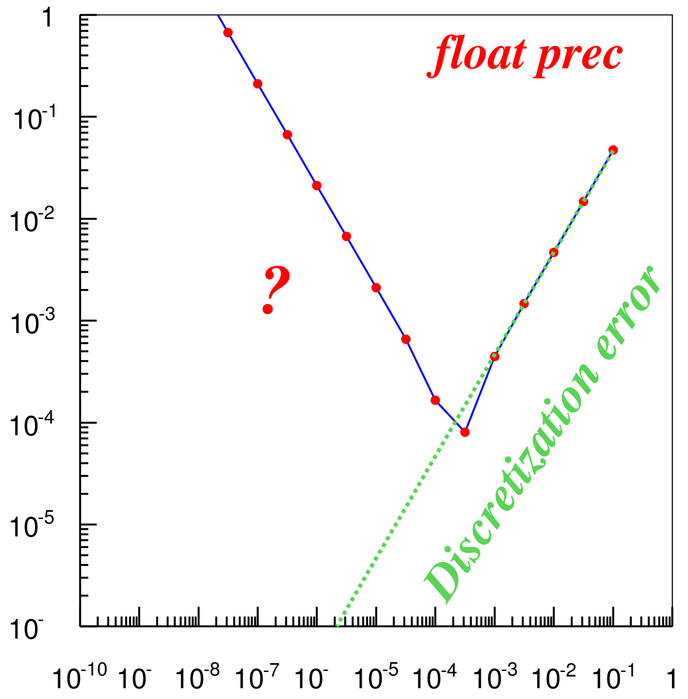
$$\Delta f_d^{'} \simeq \frac{h}{2} f^{''}(x_0)$$



Derivative computation errors

✔ Let's compute the error on the derivative $f'(x_0)$ as function of the discretization distance h : $\Delta f' = f'(x_0) - cos(x_0)$

h	$\Delta f'$
1×10^{-1}	4.7×10^{-2}
3.16×10^{-2}	1.47×10^{-2}
1×10^{-2}	4.66×10^{-3}
3.16×10^{-3}	1.46×10^{-3}
1×10^{-3}	4.44×10^{-4}
3.16×10^{-4}	8.03×10^{-5}
1×10^{-4}	1.65×10^{-4}
3.16×10^{-5}	6.55×10^{-4}



Loss of precision

✔ Every real number **x** has a machine representation $\mathbf{x_c}$

$x_c = x(1 + \varepsilon_x)$

where $|\varepsilon_x| \leq \varepsilon_M$ is the relative error associated to the machine precision

$\sim 10^{-7}$ for single precision representation

$\sim 10^{-16}$ for doubleprecision representation

two numbers subtraction

$$a_c = b_c - c_c = b(1+\varepsilon_b) - c(1+\varepsilon_c)$$
$$= \underbrace{(b-c)}_{a} + b\varepsilon_b - c\varepsilon_c$$
$$\frac{a_c}{a} = 1 + \frac{b}{a}\varepsilon_b - \frac{c}{a}\varepsilon_c$$
$$\varepsilon_a \simeq \frac{b}{a}(\varepsilon_b - \varepsilon_c) \qquad (b \simeq c)$$

two numbers multiplication

$$a_c = b_c \times c_c$$
$$= b(1+\varepsilon_b) \times c(1+\varepsilon_c)$$
$$\simeq \underbrace{bc}_{a} + bc\varepsilon_b + bc\varepsilon_c$$
$$\frac{a_c}{a} = 1 + \varepsilon_b + \varepsilon_c$$
$$\varepsilon_a \simeq \varepsilon_b + \varepsilon_c$$

Derivative computation errors (cont.)

errors

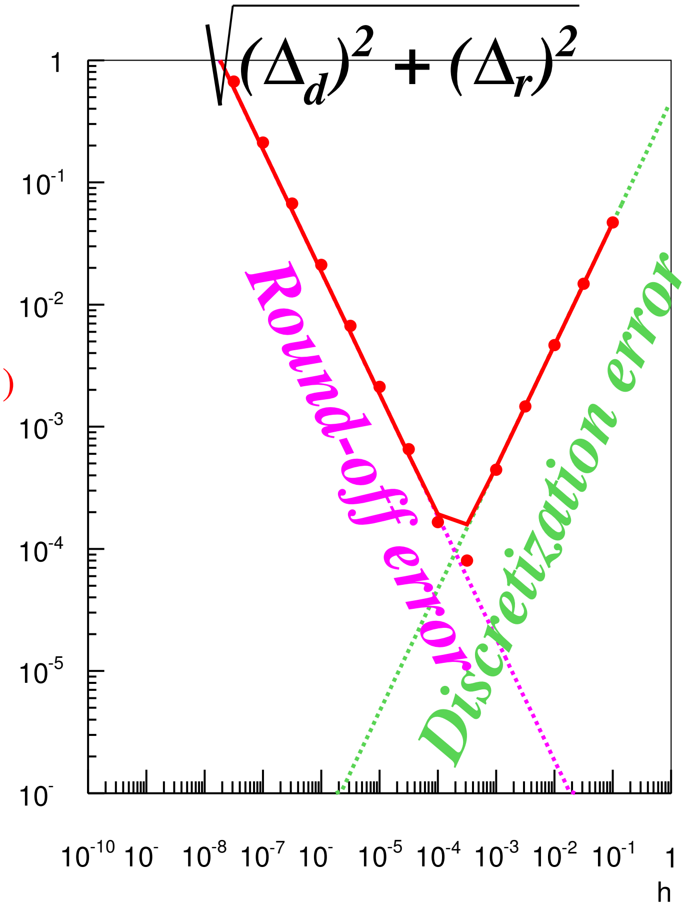
- ✔ Discretization :

  $\Delta f'_d = \frac{h}{2} f''(x_0)$

- ✔ Round-off :

$$\delta f = f(x_0 + h) - f(x_0)$$

$$\Delta f'_r = \frac{\Delta(\delta f)}{h} = f(x_0)\frac{\varepsilon_M}{h} \sim \frac{10^{-7}}{h}$$



Error sum

- ✔ The total error if a sequence of N arithmetic operations are made, can be estimated assuming **uncorrelated errors**

$$F = \sum_{i=1}^{N} x$$

$$(\Delta_F)^2 = \sum_{N} (\Delta_x)^2 = N(\Delta_x)^2$$

$$\Delta_F = \sqrt{N}\Delta_x$$

math.h constants

- ✔ Solving problems with a computer and requiring a good precision implies the use of double-precision in numbers representation
  - unless you are short in computer memory !
- ✔ a set of mathematical constants are already defined in the unix operating system
  - file : **/usr/include/math.h** (double-precision !)

```
/* Some useful constants.  */
#if defined __USE_BSD || defined __USE_XOPEN
# define M_E        2.7182818284590452354   /* e */
# define M_LOG2E    1.4426950408889634074   /* log_2 e */
# define M_LOG10E   0.43429448190325182765  /* log_10 e */
# define M_LN2      0.69314718055994530942  /* log_e 2 */
# define M_LN10     2.30258509299404568402  /* log_e 10 */
# define M_PI       3.14159265358979323846  /* pi */
# define M_PI_2     1.57079632679489661923  /* pi/2 */
# define M_PI_4     0.78539816339744830962  /* pi/4 */
# define M_1_PI     0.31830988618379067154  /* 1/pi */
# define M_2_PI     0.63661977236758134308  /* 2/pi */
# define M_2_SQRTPI 1.12837916709551257390  /* 2/sqrt(pi) */
# define M_SQRT2    1.41421356237309504880  /* sqrt(2) */
# define M_SQRT1_2  0.70710678118654752440  /* 1/sqrt(2) */
#endif
```


The hexadecimal system

- ✔ Binary numbers can be arranged in groups of 4-bits

  $(b_3\ b_2\ b_1\ b_0)_2 = b_0\ 2^0 + b_1\ 2^1 + b_2\ 2^2 + b_3\ 2^3$

  min : $(0000)_2 = 0$

  max : $(1111)_2 = 15$

  base-16 system : $0, 2, 3, 4, 5, ..., 9, A, B, C, D, E, F$
- ✔ one byte (8-bits) is represented by two hexadecimal numbers
- ✔ Examples :

  $(10\ 1111)_2 = (2F)_{16}$

  The corresponding decimal value :

  $2 \times 16^1 + 15 \times 16^0 = 47$

  $1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 47$

Characters representation

- ✔ Characters are 8-bit (byte) numbers
- ✔ ASCII (American Standard Code for Information Interchange) convention
  128 characters are represented by numerical values in the range 0-127
  7 bits needed
- ✔ The extended ASCII character set (ECS) includes 128 additional characters encoded by integers in the range 128-254
  8 bits required

Characters representation (cont.)

The ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
00	00	NUL	32	20	SP	64	40	@	96	60	‘
01	01	SOH	33	21	!	65	41	A	97	61	a
02	02	STX	34	22	"	66	42	B	98	62	b
03	03	ETX	35	23	#	67	43	C	99	63	c
04	04	EOT	36	24	$	68	44	D	100	64	d
05	05	ENQ	37	25	%	69	45	E	101	65	e
06	06	ACK	38	26	&	70	46	F	102	66	f
07	07	BEL	39	27	’	71	47	G	103	67	g
08	08	BS	40	28	(	72	48	H	104	68	h
09	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m

• • •