UNIVERSIDAD NACIONAL DE ASUNCIÓN

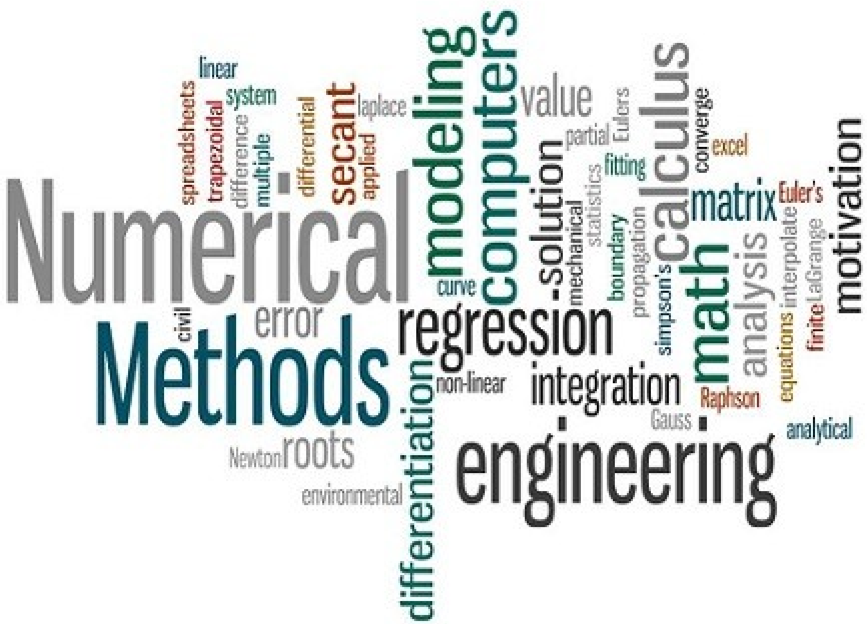
FACULTAD POLITÉCNICA

DEPARTAMENTO DE INVESTIGACIÓN, POSTGRADO Y EXTENSIÓN

Postgrado en Ciencias de la Computación

METODOS NUMÉRICOS

Prof Diego Stalder



Objetivos

- Conocer los fundamentos de matemática computacional y métodos numéricos utilizados para obtener soluciones aproximadas a problemas científicos que de otro modo serían intratables.
- Aplicar métodos numéricos para resolver problemas científicos y de ingeniería.
- Utilizar métodos como la interpolación, la diferenciación, la integración, la solución de ecuaciones lineales y no lineales, y la solución de ecuaciones diferenciales para obtener soluciones a problemas reales.
- Analizar y evaluar la precisión y exactitud de los métodos numéricos estudiados.
- Implementar métodos numéricos utilizando Python, Matlab, Fortran o C/C++.
- Escribir rutinas o programas computacionales para presentar los resultados numéricos de manera informativa.

Referencias bibliográficas

- Chapra y Canale. Numerical Methods for Engineers (6 Edition)
- Conte y Boor. Elementary Numerical Analysis: An Algorithmic Approach
- Joe D. Hoffman. Numerical Methods for Engineers and Scientists
- Jaan Kiusalaas. Numerical Methods In Engineering With Python 3.
- O'Leary D.P. Scientific Computing with Case Studies (SIAM, 2008).

Contenido(18 clases de 2hrs+1 parcial + 1 final)

I. Aproximaciones y Fuentes de Error (2 clases)

Exactitud, Precisión, Errores, Redondeo y Truncamiento, Aritmética de punto flotante.

II. Raíces de Ecuaciones No Lineales (2 clases)

Busqueda Incremental, Bisección, Newton Raphson, Secante.

III. Sistemas Lineales de Ecuaciones Algebraicas (3 clases)

Propiedades de las Matrices, Pivoteo, Triangularización, Eliminación Gaussiana, LU, QR

IV. Aproximación de funciones (3 clases)

Mínimos Cuadrados e Interpolación, Spline, Lagrange, Optimización.

V. Integración y diferenciación numérica (3 clases)

Simpsons, Roemberg, Cuadratura Gaussiana y Montecarlo.

VI. Ecuaciones Diferenciales Ordinarias (3 clases)

Taylor Series, Runge Kutta, Problema de valor inicial, Condiciones de frontera.

VII. Ecuaciones Diferenciales Parciales(2 clases)

Tipo de Ecuaciones diferenciales parciales, diferencias finitas, elementos finitos y volumen finitos.

Evaluaciones

40% Trabajo Prácticos.

30% Examen Parcial (11 de Abril, hasta el capítulo IV)

30% Examen Final (10 de Mayo)

MARZO

DOM	LUN	MAR	MIE	JUE	VIE	SAB
				1 (TJ)	2	
	5 (MN)	6	7 (TJ)	8 (TJ)	9 (AL)	
	12 (MN)	13 (AL)	14 (TJ)(MN)	15 (TJ)	16 (MN)	
	19 (MN)	20 (AL)	21 (MN)	22	23 (MN)	
	26	27 (AL)	Asueto	Feriado	Feriado	

ABRIL

DOM	LUN	MAR	MIE	JUE	VIE	SAB
	2 (MN)	3 (AL)	4 (MN)	5 (AL)	6	
	9 (MN)	10 (AL)	11 (MN)	12 (AL)	13	
	16 (MN)	17 (AL)	18 (MN)	19 (AL)	20	
	23 (MN)	24 (AL)	25 (MN)	26 (AL)	27	
	30					

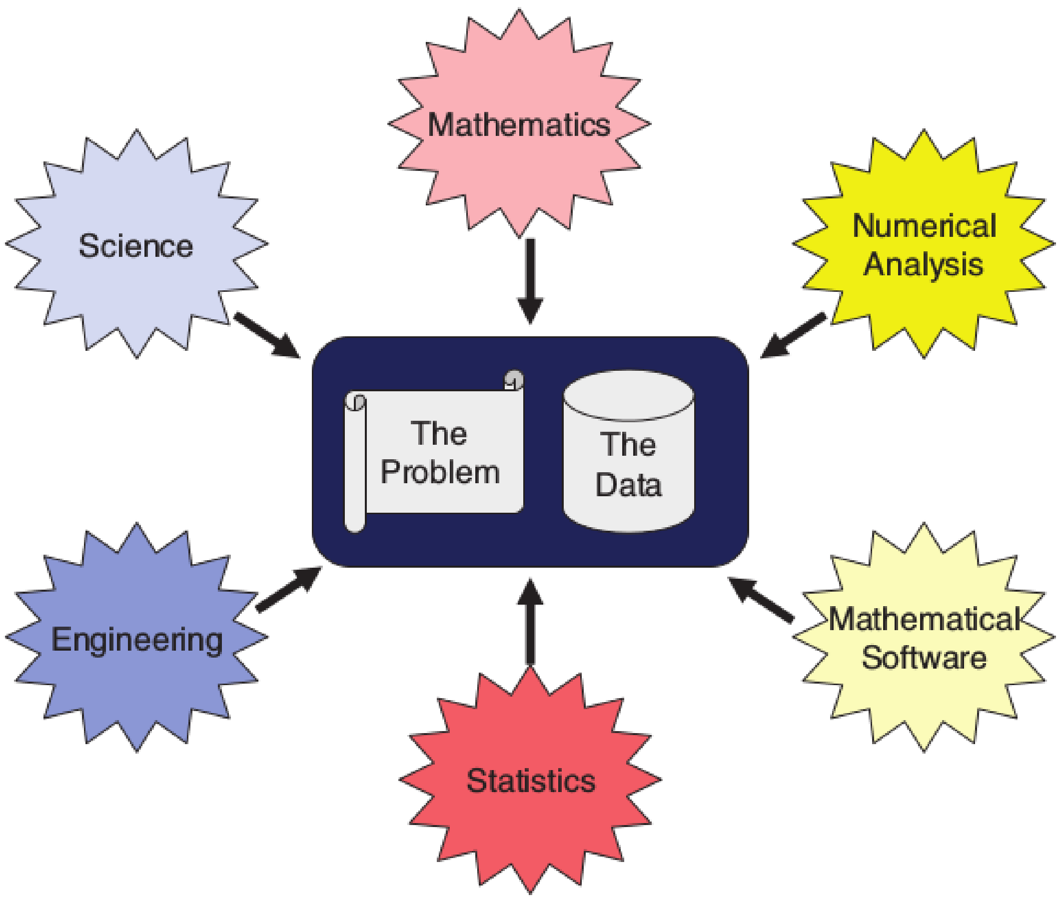
MAYO

DOM	LUN	MAR	MIE	JUE	VIE	SAB
		Feriado	2 (MN)	3 (AL)	4	
	7 (MN)	8 (MN)	9 (MN)	10 (MN)	11 (ALG)	
	Feriado	Feriado	16 (ALG)	17	18 (ALG)	
	21 (ALG)	22 (IS)	23 (ALG)	24 (IS)	26 (ALG)	
	28 (ALG)	29 (IS)	30 (ALG)	31 (IS)		

JUNIO

DOM	LUN	MAR	MIE	JUE	VIE	SAB
					1 (ALG)	
	4 (ALG)	5 (IS)	6 (ALG)	7 (IS)	8 (ALG)	
	11 (ALG)	9 (IS)	10 (ALG)	11 (IS)	12 (ALG)	
	18 (ALG)	13 (IS)	14 (ALG)	15 (IS)	16 (ALG)	
	25 (ALG)	26 (IS)	27 (ALG)	28 (IS)	29	

Ciencias Computacionales



Analista Numérico vs Científico Computacional

-- design algorithms and analyze them.

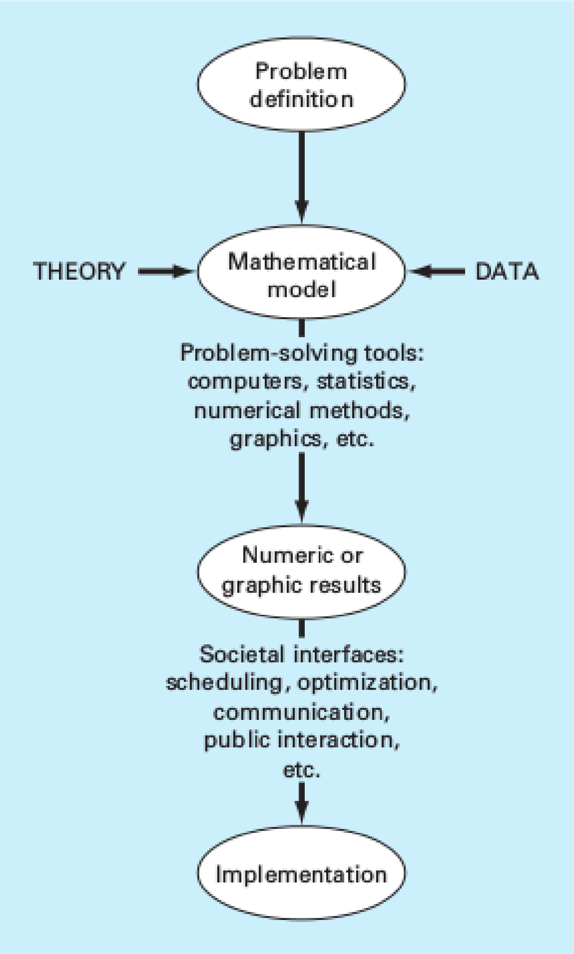
-- develop mathematical software.

-- answer questions about how accurate the final answer is.

-- works as part of an interdisciplinary team.

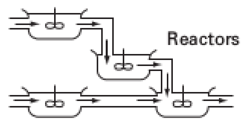
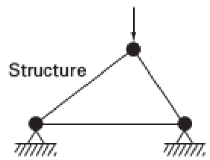
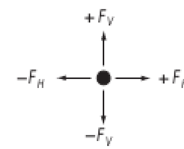
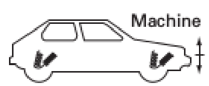
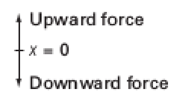
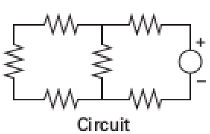
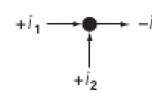
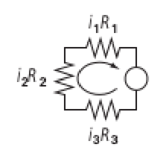
-- intelligently uses mathematical software to analyze mathematical models.

El problema

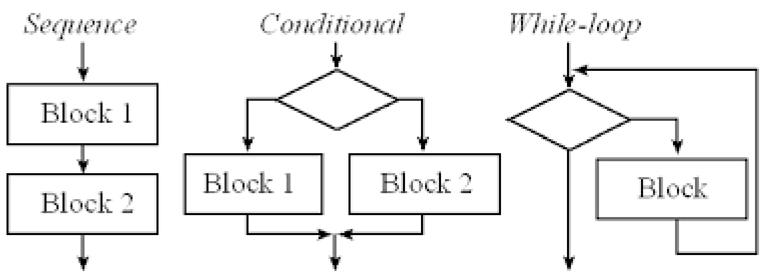


$$\text{Dependent variable} = f\left(\begin{matrix}\text{independent}\\ \text{variables}\end{matrix}, \text{parameters}, \begin{matrix}\text{forcing}\\ \text{functions}\end{matrix}\right)$$

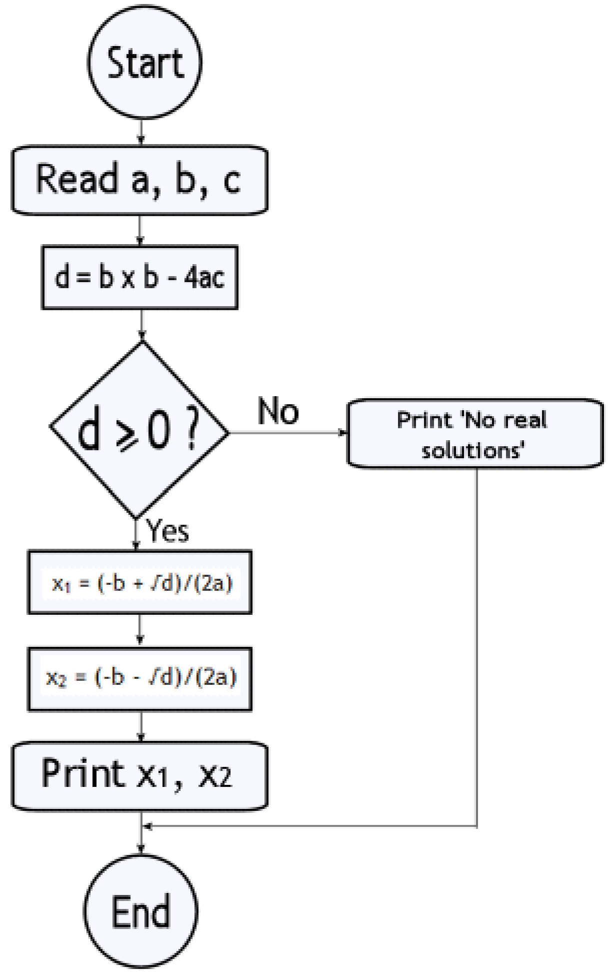
Modelos Matemáticos

Field	Device	Organizing Principle	Mathematical Expression
Chemical engineering		Conservation of mass	Mass balance:  Over a unit of time period Δmass = inputs − outputs
Civil engineering		Conservation of momentum	Force balance:  At each node Σ horizontal forces (F_H) = 0 Σ vertical forces (F_V) = 0
Mechanical engineering		Conservation of momentum	Force balance:  $m\dfrac{d^2x}{dt^2}$ = downward force − upward force
Electrical engineering		Conservation of charge	Current balance:  For each node Σ current (i) = 0
		Conservation of energy	Voltage balance:  Around each loop Σ emf's − Σ voltage drops for resistors = 0 $\Sigma\,\xi - \Sigma\, iR = 0$

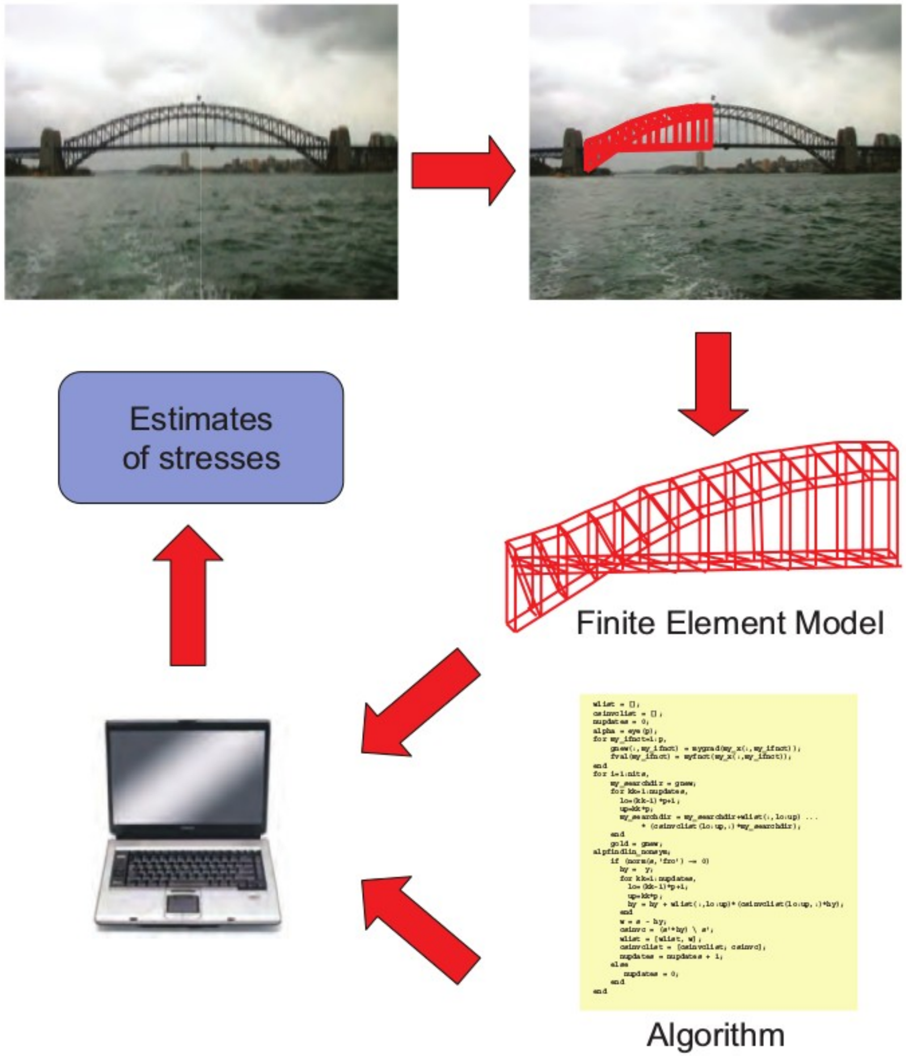
Programación




```
from math import sqrt

def main():
    print("This program solves a quadratic equation")
    print(”in the form ax^2 + bx + c = 0.")
    a = float(input("Enter the coefficient a: "))
    b = float(input("Enter the coefficeint b: "))
    c = float(input("Enter the coefficient c: "))

    disc = b**2 - 4 * a * c

    if disc < 0:
        print("There is no solution!")
    else:
        rt = sqrt(disc)
        x1 = (-b + rt)/(2 * a)
        x2 = (-b - rt)/(2 * a)
        print("The two solutions are: ", x1, x2)

main()
```



Fuentes de Error



- The engineer did not misread the measurement device.
- The model was a good approximation to the true bridge.
- The programmer did not type the value of π incorrectly.
- The computer worked flawlessly.

- Medida.
- Modelo.
- Truncamiento.
- Redondeo.

Como el computador hace los cálculos?

- Aritmética de punto fijo.

Each word (storage location) in a machine contains a fixed number of digits. (Base 10 o Base 2?, Signos?)

0	0	1	9	8	5

- Aritmética de punto flotante.

If we wanted to store 15×2^{11} , we would need 16 bits:

0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Instead, let's agree to code numbers as **two** fixed point binary numbers:

$z \times 2^{p}$,   with z = 15 saved as 01111
and p = 11 saved as 01011.

Aritmética de Punto Flotante

$$
\begin{aligned}
123.432 &= 12.3432 \text{ x } 10 \\
123.432 &= 0.123432 \text{ x } 10^3 \\
123.432 &= 12343.2 \text{ x } 10^{-2} \\
123.432 &= 0.00123432 \text{ x } 10^5
\end{aligned}
$$

Typically, each number is stored in a computer word consisting of 32 binary digits (bits) with values 0 and 1

IEEE Standard Floating Point Arithmetic

$$x = \pm z \times 2^p,$$

$$12.3432 > 1$$

$$12343.2 > 1$$

$$0.00123432 < 0.1$$

$$0.1 < 0.123432 < 1$$

z is called the mantissa

$$1 \times 2^2 = 4 \times 2^0 = 8 \times 2^{-1} \qquad 1 \leq z < 2$$

Aritmética de Punto Flotante

Computer			
IBM 7094	2	27	2^7
Burroughs 5000 Series	8	13	2^6
IBM 360/370	16	6	2^6
CDC 6000 and Cyber Series	2	48	2^{10}
DEC 11/780 VAX	2	24	2^7
Hewlett Packard 67	10	10	99

On most machines today,

single precision: $d = 24$, $m = -126$, $M = 127$

double precision: $d = 53$, $m = -1022$, $M = 1023$.

Aritmética de Punto Flotante

Single-precision numbers, 24 digits are used to represent the mantissa, and the exponent is restricted to the range $-126 \le p \le 127$.

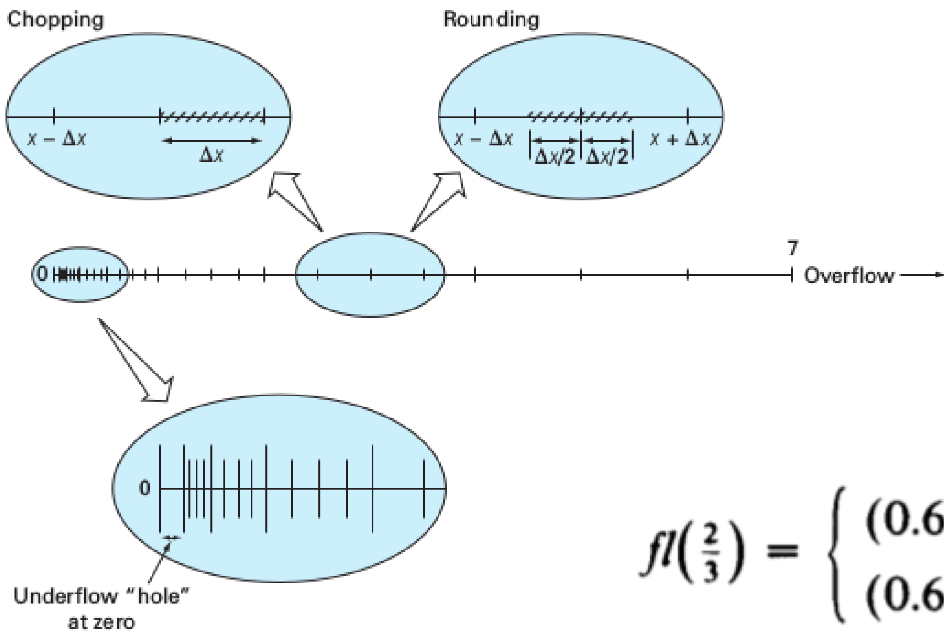
This allows us to represent numbers as close to zero as $2^{-126} \approx 1.18 \times 10^{-38}$ and as far as almost $2^{128} \approx 3.40 \times 10^{3}$

Double-precision numbers, stored in two words, using 53 digits for the mantissa, with an exponent $-1022 \le p \le 1023$.

If we perform a computation in which the exponent of the answer is outside the allowed range, we have a more or less serious error.

- Overflow.
- Underflow.
- Not-a-number.

Errores de truncamiento y redondeo



$$fl\left(\tfrac{2}{3}\right) = \begin{cases} (0.67)10^0 & \text{rounded} \\ (0.66)10^0 & \text{chopped} \end{cases}$$

$$fl(-838) = \begin{cases} -(0.84)10^3 & \text{rounded} \\ -(0.83)10^3 & \text{chopped} \end{cases}$$

Errores de truncamiento y redondeo

For normalized floating-point numbers, this proportionality can be expressed, for cases where chopping is employed, as

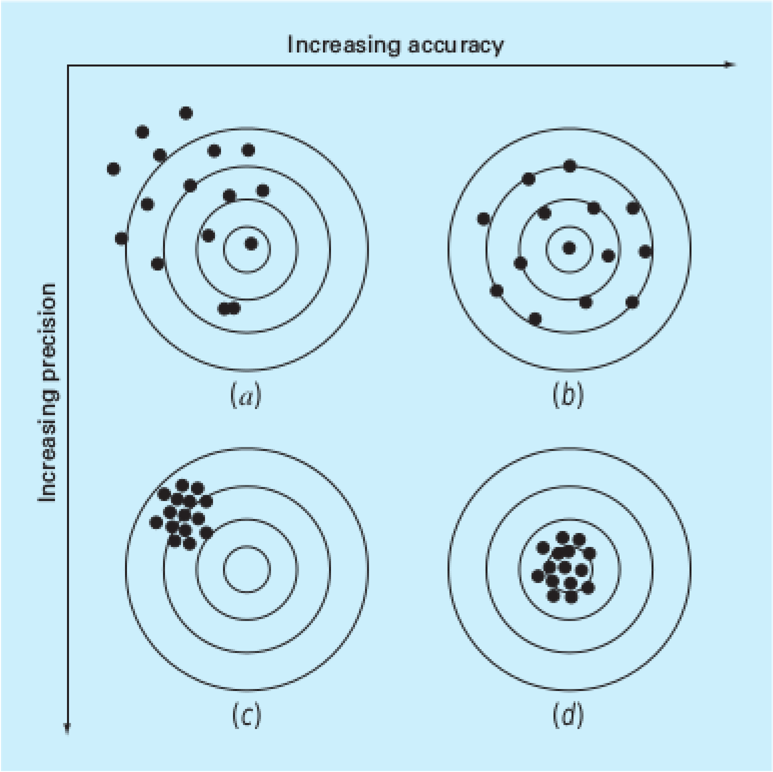
$$\frac{|\Delta x|}{|x|} \leq \mathscr{E}$$

and, for cases where rounding is employed, as

$$\frac{|\Delta x|}{|x|} \leq \frac{\mathscr{E}}{2}$$

$$\mathscr{E} = b^{1-t}$$

Exactitud vs Precisión



Como medimos el error?

Absolute error in c as an approximation to x:

$$|x - c|$$

Relative error in c as an approximation to nonzero x:

$$\frac{|x - c|}{|x|}$$

Propagación de Errores

Errors can be magnified during computation.

Example: 2.003×10^0 (suppose $\pm .001$ or .05% error)
$- 2.000 \times 10^0$ (suppose $\pm .001$ or .05% error)

Result of subtraction:

$$0.003 \times 10^0$$

but true answer could be as small as $2.002 - 2.001 = 0.001$,
or as large as $2.004 - 1.999 = 0.005$!

Propagación de Errores

función	Error	Error de la función
$q = x \pm y$	$x \pm \Delta x,$ $y \pm \Delta y$	$\Delta q = \Delta x + \Delta y$ $\Delta q = \sqrt{\Delta x^2 + \Delta y^2}$
$q = xy$	$x \pm \Delta x,$ $y \pm \Delta y$	$\frac{\Delta q}{\lvert q\rvert} = \frac{\Delta x}{\lvert x\rvert} + \frac{\Delta y}{\lvert y\rvert}$
$q = \frac{x}{y}$	$x \pm \Delta x,$ $y \pm \Delta y$	$\frac{\Delta q}{\lvert q\rvert} = \frac{\Delta x}{\lvert x\rvert} + \frac{\Delta y}{\lvert y\rvert}$
$q = Ax$	$x \pm \Delta x$	$\Delta q = \lvert A\rvert \Delta x$
$q = x^n$	$x \pm \Delta x$	$\frac{\Delta q}{\lvert q\rvert} = n\frac{\Delta x}{\lvert x\rvert}$
$q = Ax^n y^m$	$x \pm \Delta x$ $y \pm \Delta y$	$\frac{\Delta q}{\lvert q\rvert} = \sqrt{\left(n\frac{\Delta x}{\lvert x\rvert}\right)^2 + \left(m\frac{\Delta y}{\lvert y\rvert}\right)^2}$
$q = f(x)$	$x \pm \Delta x$	$\Delta q = \left\lvert\frac{df(x)}{dx}\right\rvert \Delta x$
$q = f(x,y)$	$x \pm \Delta x,$ $y \pm \Delta y$	$\Delta q = \left\lvert\frac{\partial f}{\partial x}\right\rvert \Delta x + \left\lvert\frac{\partial f}{\partial y}\right\rvert \Delta y$
$q = f(x,y,\ldots,w)$	$x \pm \Delta x,$ $y \pm \Delta y,$ $w \pm \Delta w$	$\delta q = \sqrt{\left(\frac{\partial f}{\partial x}\Delta x\right)^2 + \left(\frac{\partial f}{\partial y}\Delta y\right)^2 + \left(\frac{\partial f}{\partial w}\Delta w\right)^2}$

https://www.uv.es/zuniga/3.2_Propagacion_de_errores.pdf

Como se puede evitar la propagación de errores?

Example: Find the roots of $x^2 - 56x + 1 = 0.$

Usual algorithm: $x_1 = 28 + \text{sqrt}(783) = 28 + 27.982 \quad (\pm .0005)$

$= 55.982 \quad (\pm .0005)$

$x_2 = 28 - \text{sqrt}(783) = 28 - 27.982 \quad (\pm .0005)$

$= 0.018 \quad (\pm .0005)$

Como se puede evitar la propagación de errores?

1. Usar formulas alternativas

$$x_2 = 1 / x_1 = 0.0178628844986$$

2. Reescribir la formula.

$$\text{sqrt}(x + e) - \text{sqrt}(x) = (\text{sqrt}(x+e) - \text{sqrt}(x)) \frac{(\text{sqrt}(x+e) + \text{sqrt}(x))}{(\text{sqrt}(x+e) + \text{sqrt}(x))}$$

$$= \frac{x + e - x}{\text{sqrt}(x+e) + \text{sqrt}(x)} = \frac{e}{\text{sqrt}(x+e) + \text{sqrt}(x)}$$

so $x_2 = 28 - \text{sqrt}(783) = \text{sqrt}(784) - \text{sqrt}(783)$.

3. Usar series de Taylor

Let $f(x) = \text{sqrt}(x)$. Then

$$f(x+a) - f(x) = f\,'(x)\, a + 1/2\ f\,''(x)\, a^2 + \ldots$$

Introducción a Python

Python is an object-oriented language that was developed in the late 1980s as a scripting language (the name is derived from the British television series, Monty Python's Flying Circus)

Python programs are not compiled into machine code, but are run by an interpreter.

The great advantage of an interpreted language is that programs can be tested and debugged quickly, allowing the user to concentrate more on the principles behind the program and less on the programming itself.

Python is an open-source software, which means that it is free.

Aritmética de Punto Flotante

Data type	Description
`bool_`	Boolean (True or False) stored as a byte
`int_`	Default integer type (same as C `long`; normally either `int64` or `int32`)
intc	Identical to C `int` (normally `int32` or `int64`)
intp	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
`complex_`	Shorthand for `complex128`.
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

Additionally to `intc` the platform dependent C integer types `short`, `long`, `longlong` and their unsigned versions are defined.