

Numerical Analysis

João Janela

`jjanela@iseg.utl.pt`

version: (March 1, 2015)

Spring 2015

Goals

- Introduction to scientific computing and numerical analysis.
- Address real cases of mathematical modelling and computer simulation.

Assessment

- Assignments during the semester (30 %)
- Final Exam (70 %)

Office Hours

- Every Monday, 14:00 - 16:00
- For other schedules contact me by email
(jjanela@iseg.ulisboa.pt)

Syllabus

- Introduction to scientific computing software (Mathematica)
- Conditioning and stability of numerical algorithms
- Numerical methods for nonlinear equations and systems of nonlinear equations. Bisection method, fixed point method and Newton's method. Equations in infinite dimensional spaces.
- Numerical methods for linear systems. Gaussian elimination and triangular factorisations. Iterative methods of Jacobi, Gauss-Seidel and SOR.

Syllabus (cont.)

- Numerical unconstrained optimisation. Differential and non-differential methods.
- Functional interpolation and approximation.
- Numerical derivation and integration.
- Numerical methods for initial value problems.

Wolfram Mathematica



- Using the program
- Rules and Syntax
- Examples

Some History...

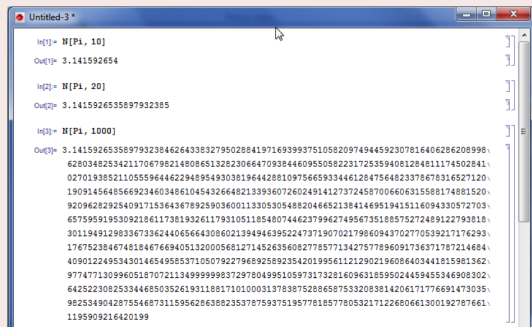
- The software was designed by Stephen Wolfram and his team in the 80's
- The first version was launched in 1988 and spread quickly among the scientific community.
- Initially used by mathematicians and physicists but soon used in numerous other areas.
- Since the very beginning was made available for multiple platforms(Windows, Unix, Linux, MacOS, NeXt, OS2,...)



What is Mathematica?

Mathematica is an interactive symbolic and numerical calculator with its own programming language (Based on C) with a large spectrum of applications.

Numerical calculations with arbitrary precision!



```
Untitled-3 *  
  
In[1]:= N[Pi, 10]  
Out[1]= 3.141592654  
  
In[2]:= N[Pi, 20]  
Out[2]= 3.1415926535897932385  
  
In[3]:= N[Pi, 1000]  
Out[3]= 3.141592653589793238462643383279502884197169399375105820974944592307816406286208998\  
628034825342117067982148086513282306647093844609505822317253594081284811174502841\  
0270193852110555964462294895493038196442881097566593344612847564823378678316527120\  
1909145648566923460348610454326648213393607260249141273724587006606315588174881520\  
9209628292540917153643678925903600113305305488204665213841469519415116094330572703\  
657595919530921861173819326117931051185480744623799627495673518857527489122793818\  
3011949129833673362440656643086021394946395224737190702179860943702770539217176293\  
1767523846748184676694051320005681271452635608277857713427577896091736371787214684\  
4090122495343014654958537105079227968925892354201995611212902196086403441815981362\  
9774771309960518707211349999998372978049951059731732816096318595024459455346908302\  
6425223082533446850352619311881710100031378387528865875332083814206171776691473035\  
9825349042875546873115956286388235378759375195778185778053217122680661300192786761\  
1195909216420199
```

Symbolic calculations and simplification of expressions

```

In[5]:= D[x^2 + 1, x]
Out[5]= 2 x

In[7]:= D[(x + 1) / (x^2 + 3), x]
Out[7]=  $\frac{2 x (1 + x)}{(3 + x^2)^2} + \frac{1}{3 + x^2}$ 

In[8]:= D[Sin[Exp[x] + Tan[x^2 + 1]], {x, 4}]
Out[8]=  $(e^x + 2 x \operatorname{Sec}[1 + x^2])^4 \operatorname{Sin}[e^x + \operatorname{Tan}[1 + x^2]] - 6 \operatorname{Cos}[e^x + \operatorname{Tan}[1 + x^2]] (e^x + 2 x \operatorname{Sec}[1 + x^2])^2 (e^x + 2 \operatorname{Sec}[1 + x^2]^2 + 8 x^2 \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2]) - 3 \operatorname{Sin}[e^x + \operatorname{Tan}[1 + x^2]] (e^x + 2 \operatorname{Sec}[1 + x^2]^2 + 8 x^2 \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2])^2 - 4 (e^x + 2 x \operatorname{Sec}[1 + x^2])^2 \operatorname{Sin}[e^x + \operatorname{Tan}[1 + x^2]] (e^x + 16 x^3 \operatorname{Sec}[1 + x^2]^4 + 24 x \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2] + 32 x^3 \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2]^2) + \operatorname{Cos}[e^x + \operatorname{Tan}[1 + x^2]] (e^x + 96 x^2 \operatorname{Sec}[1 + x^2]^4 + 24 \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2] + 256 x^4 \operatorname{Sec}[1 + x^2]^4 \operatorname{Tan}[1 + x^2] - 192 x^2 \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2]^2 + 128 x^4 \operatorname{Sec}[1 + x^2]^2 \operatorname{Tan}[1 + x^2]^3)$ 

```


Vectors, matrices, tensors, ...

Untitled-3 *

```
In[23]:= A = Table[If[i <= j, 1, 0], {i, 1, 10}, {j, 1, 10}]; A // MatrixForm
```

Out[23]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[24]:= Inverse[A] // MatrixForm
```

Out[24]//MatrixForm=

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

I

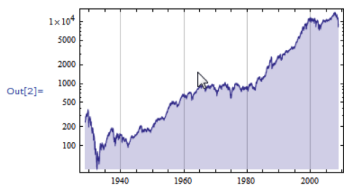
```
In[25]:= Det[A]
```

Out[25]= 1

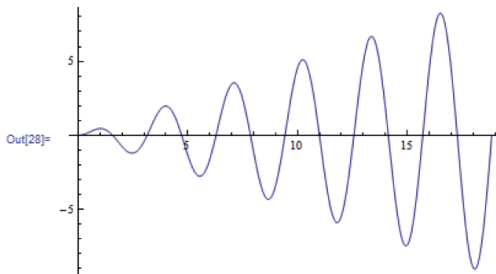
(+)

Graphics ...

```
In[2]:= DateListLogPlot[FinancialData["^DJI", All], Joined -> True, Filling -> Bottom]
```

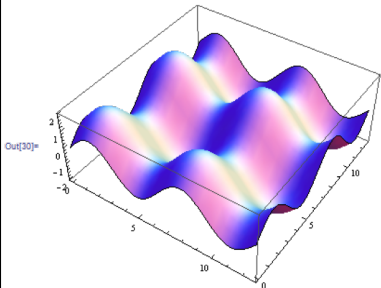


```
In[28]:= Plot[x * Sin[x] * Cos[x], {x, 0, 6 * Pi}]
```

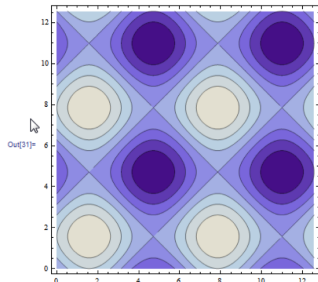


Graphics ...

```
In[30]= Plot3D[Sin[x] + Sin[y], {x, 0, 4*Pi}, {y, 0, 4*Pi}, Mesh -> False]
```



```
In[31]= ContourPlot[Sin[x] + Sin[y], {x, 0, 4*Pi}, {y, 0, 4*Pi}, Mesh -> False]
```



Equation solving

```
In[40]= Solve[2 * x^2 + a * x + 1 == 0, x]
```

```
Out[40]= {{x -> 1/4 (-a - Sqrt[-8 + a^2])}, {x -> 1/4 (-a + Sqrt[-8 + a^2])}}
```

```
In[41]= Solve[x^7 + x^6 - 5 * x + 1 == 0, x] // N
```

```
Out[41]= {{x -> -1.57723}, {x -> 0.200015}, {x -> 1.14013}, {x -> -0.862705 - 1.07939 i},  
{x -> -0.862705 + 1.07939 i}, {x -> 0.481248 - 1.1066 i}, {x -> 0.481248 + 1.1066 i}}
```

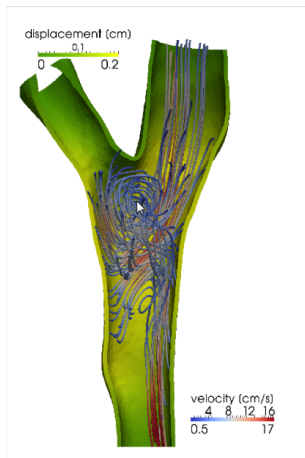
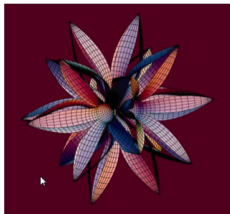
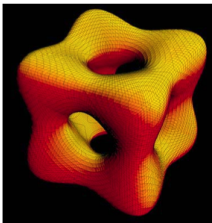
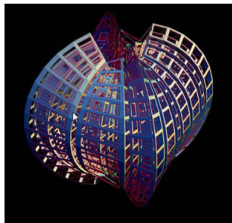
```
In[42]= Solve[{x^2 + y == 1, x + 4 * y == 0}, {x, y}]
```

```
Out[42]= {{x -> 1/8 (1 - Sqrt[65]), y -> 1/32 (-1 + Sqrt[65])}, {x -> 1/8 (1 + Sqrt[65]), y -> 1/32 (-1 - Sqrt[65])}}
```

```
In[43]= DSolve[y''[x] + 3 * y[x] == 0, y[x], x]
```

```
Out[43]= {{y[x] -> C[1] Cos[Sqrt[3] x] + C[2] Sin[Sqrt[3] x]}}
```

And much more



Structure of Mathematica

- **KERNEL:** Interprets the code, splits and launches computations, returns results, stores definitions, etc.
- **FRONTEND:** Provides an interface for code editing and results visualisation (text, graphics, movies, sound, etc).

Contains libraries with thousands of functions and modules for the solution of problems in a wide range of scientific areas.

Contains error detection and solution tools like a debugger.

Notebooks

The standard Mathematica file is called a 'Notebook'. It is composed of **cells** with different properties like **Input**, **Output**, different text styles, etc



The screenshot shows a Mathematica notebook cell with a light gray border. On the right side, there are blue closing brackets: a pair of large closing square brackets at the top, a closing square bracket in the middle, and a pair of closing curly braces at the bottom. The cell content includes:

- A bold title: **Este é um título**
- A subtitle: Este é um subtítulo
- An input line: `In[44]:= 1 + 1`
- An output line: `Out[44]:= 2`
- A small icon of a speech bubble at the bottom left.

A cell is evaluated by pressing "Enter" ou "Shift + Return".

Some basic rules

- One uses ";" in the end of a line in order to suppress the output
- Pre-defined function names always start with a capital letter (Sin, Cos, etc.)
- Function arguments always appear between square brackets (Sin[1], Cos[2], etc.)
- Keyways "{,}" are used to define vectors and lists
- Parenthesis are used to group terms like $(1 + \text{Cos}[x])\text{Sin}[x]$
- "=" is the attribution operator. (e.g. $x = 1$ attributes the value 1 to variable x)
- "==" expresses equality ($1==2$ returns False)
- ":=" is used to make a definition
- "x_" refers to an arbitrary expression denoted by x.

```
In[46]:= x = 1
```

```
Out[46]= 1
```

```
In[47]:= x + 34
```

```
Out[47]= 35
```

```
In[48]:= x == 1
```

```
Out[48]= True
```

```
In[49]:= x == 0
```

```
Out[49]= False
```



Some basic rules (cont.)

One can insert comments within the code using "(*)" and "(*)".

```
In[50]:= (* O texto que aqui aparece é um comentário, não é avaliado *)  
Sqrt[23 + Sin[45.12]]
```

```
Out[50]= 4.88955
```

```
(+)
```

Variable names can be freely chosen as long as they **i.** don't contain free space; **ii.** don't start with a number; **iii.** do not coincide with a protected name.

```
In[51]:= OlaBomDia = 23
```

```
Out[51]= 23
```

```
In[53]:= 88 BomDia = 12
```

```
Set::write : Tag Times in 88 BomDia is Protected. >>
```

```
Out[53]= 12
```

```
In[54]:= Ola Bom Dia = 12
```

```
Set::write : Tag Times in Bom Dia Ola is Protected. >>
```

```
Out[54]= 12
```

Defining new functions

```
In[64]:= f[x_] := Sin[Abs[x * Cos[x]]]
```

```
In[65]:= f[Pi]
```

```
Out[65]= 0
```

```
In[66]:= f[10] + f[20]
```

```
Out[66]= -Sin[10 Cos[10]] + Sin[20 Cos[20]]
```

```
In[67]:= f[10.] + f[20]
```

```
Out[67]= 1.81243
```

Defining new functions

```
In[64]:= f[x_] := Sin[Abs[x * Cos[x]]]
```

```
In[65]:= f[Pi]
```

```
Out[65]= 0
```

```
In[66]:= f[10] + f[20]
```

```
Out[66]= -Sin[10 Cos[10]] + Sin[20 Cos[20]]
```

```
In[67]:= f[10.] + f[20]
```

```
Out[67]= 1.81243
```

"x_" stands for a general function argument to be evaluated at run time

Control structures (If)

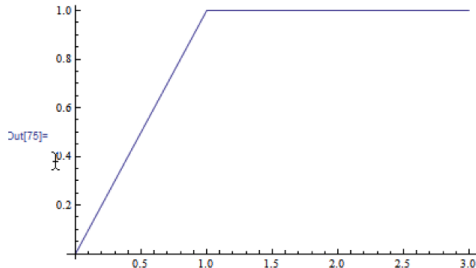
?? If

If[*condition*, *t*, *f*] gives *t* if *condition* evaluates to True, and *f* if it evaluates to False.

If[*condition*, *t*, *f*, *u*] gives *u* if *condition* evaluates to neither True nor False. \Rightarrow

```
In[74]:= g[x_] := If[x >= 1, 1, x]
```

```
In[75]:= Plot[g[x], {x, 0, 3}]
```



Control structures (Which)

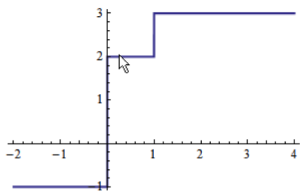
? Which

Which[*test*₁, *value*₁, *test*₂, *value*₂, ...] evaluates each of the *test*_{*i*} in turn, returning the value of the *value*_{*i*} corresponding to the first one that yields True. >>

```
In[84]= h[x_] := Which[x ≤ 0, -1,  
    0 < x < 1, 2,  
    x ≥ 1, 3]
```

```
In[85]= Plot[h[x], {x, -2, 4}]
```

Out[85]=



Repetition structures (Do)

? Do

`Do[expr, {i, imax}]` evaluates *expr* *i*_{max} times.

`Do[expr, {i, imax}]` evaluates *expr* with the variable

i successively taking on the values 1 through *i*_{max} (in steps of 1).

`Do[expr, {i, imin, imax}]` starts with *i* = *i*_{min}.

`Do[expr, {i, imin, imax, di}]` uses steps *di*.

`Do[expr, {i, {i1, i2, ...}}]` uses the successive values *i*₁, *i*₂, ...

`Do[expr, {i, imin, imax}, {j, jmin, jmax}, ...]`

evaluates *expr* looping over different values of *j*, etc. for each *i*. >>

```
In[90]:= x = 1;  
         Do[x = x + i, {i, 1, 10}];  
         x
```

Repetition structures (While)

?? While

While[*test*, *body*] evaluates *test*, then *body*, repetitively, until *test* first fails to give True. >>

```
In[100]:= x = 5;  
While[x > 0,  
  x = x - 1];  
x  
Out[102]= 0
```

```
In[96]:= x = 5;  
While[x > 0,  
  x = x - 1;  
  Print[x]]  
4  
3  
2  
1  
0
```

Repetition structures (Sum)

?? Sum

Sum[f, {i, i_{max}}] evaluates the sum $\sum_{i=1}^{i_{max}} f$.

Sum[f, {i, i_{min}, i_{max}}] starts with $i = i_{min}$.

Sum[f, {i, i_{min}, i_{max}, di}] uses steps di .

Sum[f, {i, {i₁, i₂, ...}}] uses successive values i_1, i_2, \dots

Sum[f, {i, i_{min}, i_{max}}, {j, j_{min}, j_{max}}, ...] evaluates the multiple sum $\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} \dots f$.

Sum[f, i] gives the indefinite sum $\sum_i f$. \gg

```
In[147]:= Sum[i, {i, 1, 1000}]
```

```
Out[147]= 500 500
```

```
In[148]:= Sum[i^2, {i, 1, 1000}]
```

```
Out[148]= 333 833 500
```


Vectors and matrices

A vector is defined as a list of similar elements, not necessarily numbers

```
In[103]:= x = {1, 0, 34, 1, 2, 4}
```

```
Out[103]:= {1, 0, 34, 1, 2, 4}
```

```
In[104]:= x[[3]]
```

```
Out[104]:= 34
```

```
In[105]:= x[[6]]
```

```
Out[105]:= 4
```

```
In[106]:= Length[x]
```

```
Out[106]:= 6
```

```
In[107]:= x + x
```

```
Out[107]:= {2, 0, 68, 2, 4, 8}
```

```
In[109]:= 3 * x
```

```
Out[109]:= {3, 0, 102, 3, 6, 12}
```

```
In[113]:= Sin[x]
```

Choose how to enter input

```
Out[113]:= {Sin[1], 0, Sin[34], Sin[1], Sin[2], Sin[4]}
```

Vector and matrices

A matrix is defined as a list of row vectors.

```
In[120]:= A = {{1, 0, 3}, {4, 5, 6}, {7, 8, 9}};
```

```
In[118]:= MatrixForm[A]
```

```
Out[118]/MatrixForm=
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
In[123]:= Inverse[A] // MatrixForm
```

```
Out[123]/MatrixForm=
```

$$\begin{pmatrix} \frac{1}{4} & -2 & \frac{5}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{4} & \frac{2}{3} & -\frac{5}{12} \end{pmatrix}$$

```
In[121]:= Det[A]
```

```
Out[121]= -12
```

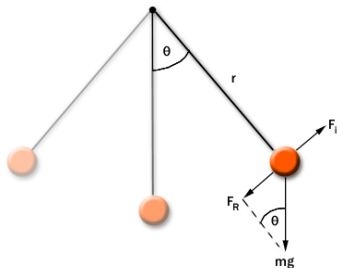
```
In[125]:= A . {1, 2, 4}
```

```
Out[125]= {13, 38, 59}
```

Sources of error in scientific computing

- Modelling errors
- Errors in model data
- Errors in number representation
- Errors produced during the solution procedure

Modelling error



The motion of a simple pendulum can be described by the ODE

$$m \frac{d^2 \theta}{dt^2} = -mg \sin \theta.$$

- This model does not take into account some relevant physical aspects like the resistance of air.
- It consists of a nonlinear differential equation, for which there are no explicit closed form solutions. If θ is small, one can use the approximation $\sin \theta \approx \theta$ obtaining the linear EDO

$$\frac{d^2 \theta}{dt^2} = -g\theta$$

Physical problem

$$\frac{d^2\theta}{dt^2} = -g \sin \theta$$

$$\frac{d^2\theta}{dt^2} = -g\theta$$

solution to the linearised problem:

$$\theta(t) = \frac{\pi\sqrt{g} \cos(\sqrt{g}t) - 4 \sin(\sqrt{g}t)}{4\sqrt{g}}$$

- How large was the error introduced by the linearisation ?
- How large was the error introduced by neglecting air resistance ?

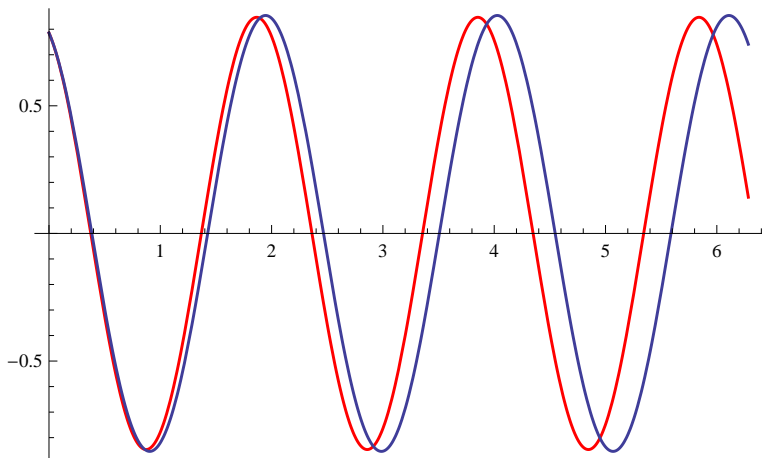


Figure: Comparison between the solution of the linear problem (red) and the (numerical) solution of the nonlinear problem (blue)

Data errors

Models often depend on parameters related to experimental data or sampling, which are inherently affected by a degree of error or uncertainty. In the case of the pendulum, depending on its location, the value of the acceleration of gravity (g), may vary.

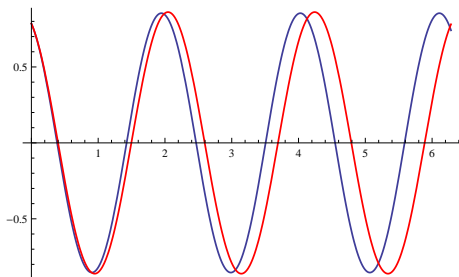


Figure: Solution obtained with $g = 10m/s^2$ (blue) and with $g = 9m/s^2$ (red).

Representation errors

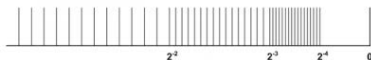
Computers can only store finite representations of real numbers, normally in a so called **floating point system**

$$\begin{aligned}
 x &= \pm \overbrace{0.a_1 a_2 \cdots a_t}^{\text{mantissa}} \times \beta^e \\
 &= \pm a_1 a_2 \cdots a_t \beta^{e-t}
 \end{aligned}$$

- β : base of the numeric system (normally 10, 2, 8, 16, ...)
- a_1, \dots, a_t : digits in mantissa $a_i \in \{0, \dots, \beta - 1\}$.
- e : Exponent.

Set of representable numbers

$$FP(\beta, t, L, U)$$



$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2} \epsilon_m, \quad \epsilon_m = \beta^{1-t}$$

Error propagation

Definition (Error)

Let $x, \hat{x} \in \mathbb{R}^n$, where \hat{x} is an approximation of x . Then we define

- Error: $e_x = \hat{x} - x$
- Absolute error: $\|e_x\| = \|\hat{x} - x\|$
- Relative error: $\varepsilon_x = \frac{\|\hat{x} - x\|}{\|x\|}$, $x \neq 0$.
- Number of significant digits: \hat{x} is said to approximate x with p significant digits if p is the largest nonnegative integer such that

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq 0.5 \times 10^{-p}.$$

Sometimes we use the approximation $\varepsilon_x \approx \frac{\|\hat{x} - x\|}{\|\hat{x}\|}$. In fact, in the one-dimensional case,

$$\frac{\hat{x} - x}{\hat{x}} = \frac{\varepsilon_x}{1 + \varepsilon_x} = \varepsilon_x (1 - \varepsilon_x + \varepsilon_x^2 - \varepsilon_x^3 + \dots) = \varepsilon_x - \varepsilon_x^2 + \varepsilon_x^3 - \dots$$

Error propagation

Propagation of errors in function computations

If we take \hat{x} instead of x , and use it as the input of a given function f , what can we say about the error committed when approximating $f(x)$ by $f(\hat{x})$?

Theorem

Let $f : I \subset \mathbb{R} \rightarrow \mathbb{R}$ be of class C^2 and $x, \hat{x} \in I$. Then

- 1 $f(\hat{x}) = f(x) + f'(x)(\hat{x} - x) + o(\hat{x} - x)$.
- 2 $e_{f(x)} = f(\hat{x}) - f(x) \approx f'(x)e_x$
- 3 If $f(x) \neq 0$ then $\varepsilon_{f(x)} = \frac{|f(\hat{x}-f(x))|}{|f(x)|} \approx \left| \frac{xf'(x)}{f(x)} \right| \varepsilon_x$

OBS: We define the condition number of f at x as

$$\text{cond}f(x) = \frac{xf'(x)}{f(x)}$$

Condition number

Definition

- A function is "hill-conditioned" near x if $\text{cond}_f(x) \gg 1$. This means that one can get large relative errors in the computed values of f , even if the argument has small relative error.
- A function is "well-conditioned" if it is not hill-conditioned :)

Remark

The extension to functions of several variables is straightforward:

$$e_{f(x)} \approx \sum_{i=1}^n \frac{\partial f}{\partial x_i}(x) e_{x_i}, \quad \varepsilon_{f(x)} \approx \sum_{i=1}^n \left| \frac{x_i \frac{\partial f}{\partial x_i}(x)}{f(x)} \right| \cdot \varepsilon_{x_i}$$

Example

Obtain error propagation formulas for the exponential and square root functions. Do the same for the sum and product of real

Propagation of error in algorithms

Suppose we want to compute values of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ using k steps:

$$z_1 = \varphi_1(x), z_2 = \varphi_2(x, z_1), \dots, z_k = \varphi_k(x, z_1, \dots, z_{k-1})$$

Naturally, if the algorithm is well conceived, we must have $z_k = f(x)$. In fact $f = \varphi_k \circ \varphi_{k-1} \circ \dots \circ \varphi_1$.

Remark

Because of the round-off errors at each step of the algorithm, what we actually have is

$$z_1 = \varphi_1(x) + \varepsilon_1$$

$$z_2 = \varphi_2(x, z_1) + \varepsilon_2$$

$$\vdots$$

$$z_k = \varphi_k(x, z_1, z_2, \dots, z_{k-1}) + \varepsilon_k$$

Stability

Computing sequentially all relative errors from the last expression we realize that

Formula for the propagation of errors in algorithms

$$\varepsilon_{z_k} = \left(\sum_{i=1}^n \text{cond}_i(f) \varepsilon_{x_i} \right) + \left(\sum_{i=1}^k Q_i(x) \varepsilon_i \right)$$

Remark

- *The first term depends only on the function and is related to the conditioning of f , which does not depend on the particular algorithm.*
- *The second term depends on the round-off errors and so it depends on the algorithm.*

Definition (Stable algorithm)

An algorithm for the computation of $f(x)$ is stable if f is well conditioned and all the coefficients $Q_i(x), i = 1, \dots, k$ are bounded in a neighbourhood of x .

Example

Compute $F = (A + B + C + D)/E$, where $A = 0.492$, $B = 0.603$, $C = -0.494$, $D = -0.602$, $E = 10^{-5}$. Use a floating point system with decimal base and three digits in the mantissa and considerer two different algorithms:

$$\textcircled{1} \quad z_1 = A + B, \quad z_2 = C + D, \quad z_3 = z_1 + z_2, \quad z_4 = z_3/E$$

$$\textcircled{2} \quad z_1 = A + C, \quad z_2 = B + D, \quad z_3 = z_1 + z_2, \quad z_4 = z_3/E$$

Example

Consider the functions

$$f(x) = \frac{x}{1 - \sqrt{1-x}}, x \neq 0 \quad g(x) = 1 + \sqrt{1-x}$$

Since they are in fact equal for $x \neq 0$, compare the underlying algorithms.

Algorithm 1

$$\begin{aligned}z_1 &= 1 - x \\z_2 &= \sqrt{z_1} \\z_3 &= 1 - z_2 \\z_4 &= \frac{x}{z_3}\end{aligned}$$

Algorithm 2

$$\begin{aligned}z_1 &= 1 - x \\z_2 &= \sqrt{z_1} \\z_3 &= 1 + z_2\end{aligned}$$

Analysis of Algorithm 1

$$\varepsilon_{z_1} = \frac{x(-1)}{1-x} \varepsilon_x + \varepsilon_{a_1} = \frac{x}{x-1} \varepsilon_x + \varepsilon_{a_1}$$

$$\varepsilon_{z_2} = \frac{z_1 \cdot \frac{1}{2\sqrt{z_1}}}{\sqrt{z_1}} + \varepsilon_{a_2} = \frac{1}{2} \varepsilon_{z_1} + \varepsilon_{a_2} = \frac{x}{2(x-1)} \varepsilon_x + \frac{1}{2} \varepsilon_{a_1} + \varepsilon_{a_2}$$

$$\begin{aligned} \varepsilon_{z_3} &= \frac{z_2 \cdot (-1)}{1-z_2} \varepsilon_{z_2} + \varepsilon_{a_3} = \\ &= -\frac{x\sqrt{1-x}}{2(x-1)(1-\sqrt{1-x})} \varepsilon_x - \frac{\sqrt{1-x}}{2(1-\sqrt{1-x})} \varepsilon_{a_1} - \frac{\sqrt{1-x}}{1-\sqrt{1-x}} \varepsilon_{a_2} \\ &\quad + \varepsilon_{a_3} \end{aligned}$$

$$\varepsilon_{z_4} = \dots = \varepsilon_x - \varepsilon_{z_3}$$

Finally,

$$\begin{aligned} \varepsilon_{z_4} &= \underbrace{\left(1 - \frac{x\sqrt{1-x}}{2(x-1)(1-\sqrt{1-x})}\right)}_{\text{cond } f(x)} \varepsilon_x \underbrace{- \frac{\sqrt{1-x}}{2(1-\sqrt{1-x})}}_{Q_1(x)} \varepsilon_{a_1} \\ &\quad - \underbrace{\frac{\sqrt{1-x}}{1-\sqrt{1-x}}}_{Q_2(x)} \varepsilon_{a_2} + \varepsilon_{a_3} \\ &= \text{cond}_x(f) \varepsilon_x + Q_1(x) \varepsilon_{a_1} + Q_2(x) \varepsilon_{a_2} + Q_3(x) \varepsilon_{a_3} \end{aligned}$$

The condition number is bounded in a neighbourhood of $x = 0$ but $Q_1(x)$ and $Q_2(x)$ unbounded. So f is well-conditioned but the algorithm is not stable.

Algorithm 1 is unstable. And so what?

i	$f(10^{-i})$	$g(10^{-i})$
5	1.99999	1.99999
7	2.	2.
9	2.	2.
11	2.	2.
12	1.99982	2.
13	1.99716	2.
14	2.0016	2.
15	1.80144	2.
16	0.90072	2.
17	∞	2.

In this example the relative error associated to the computation of $f(10^{-16})$ is $\frac{2.-0.90072}{2} \times 100\% \approx 55\%$.