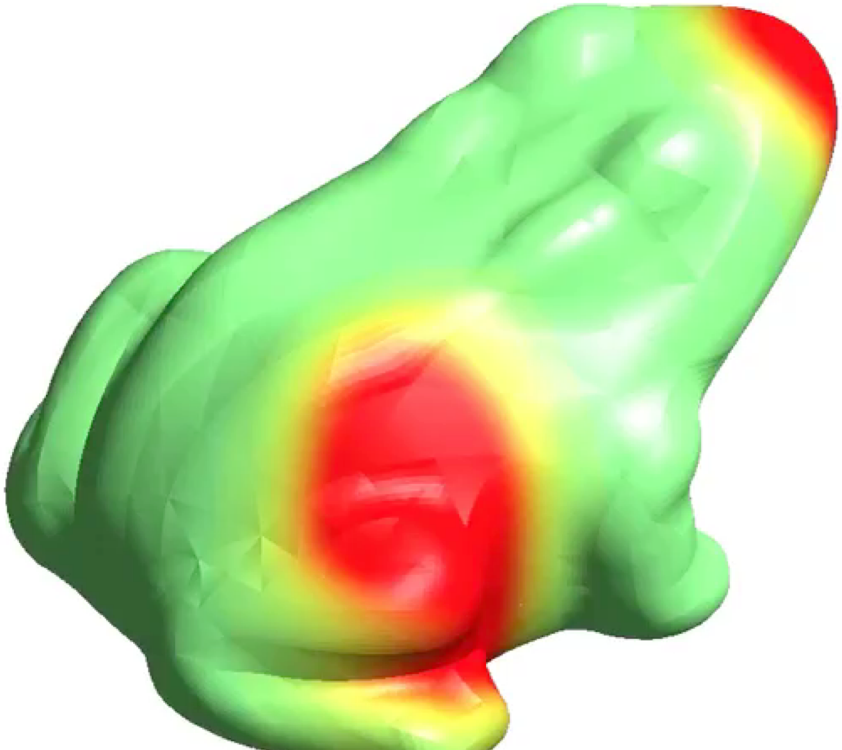
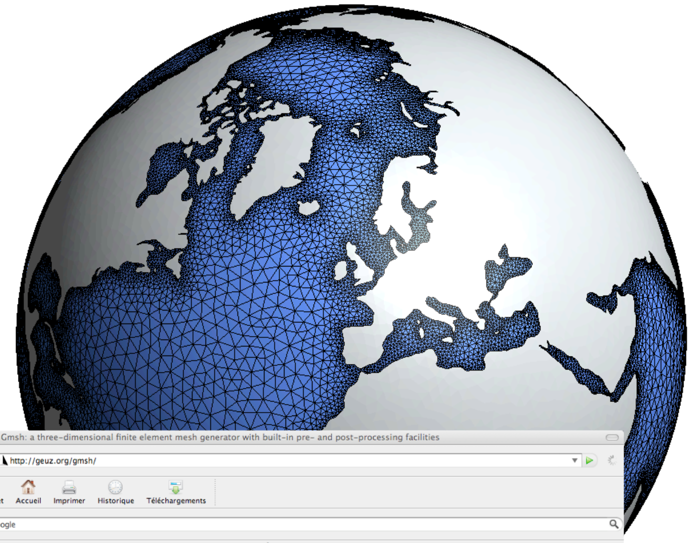
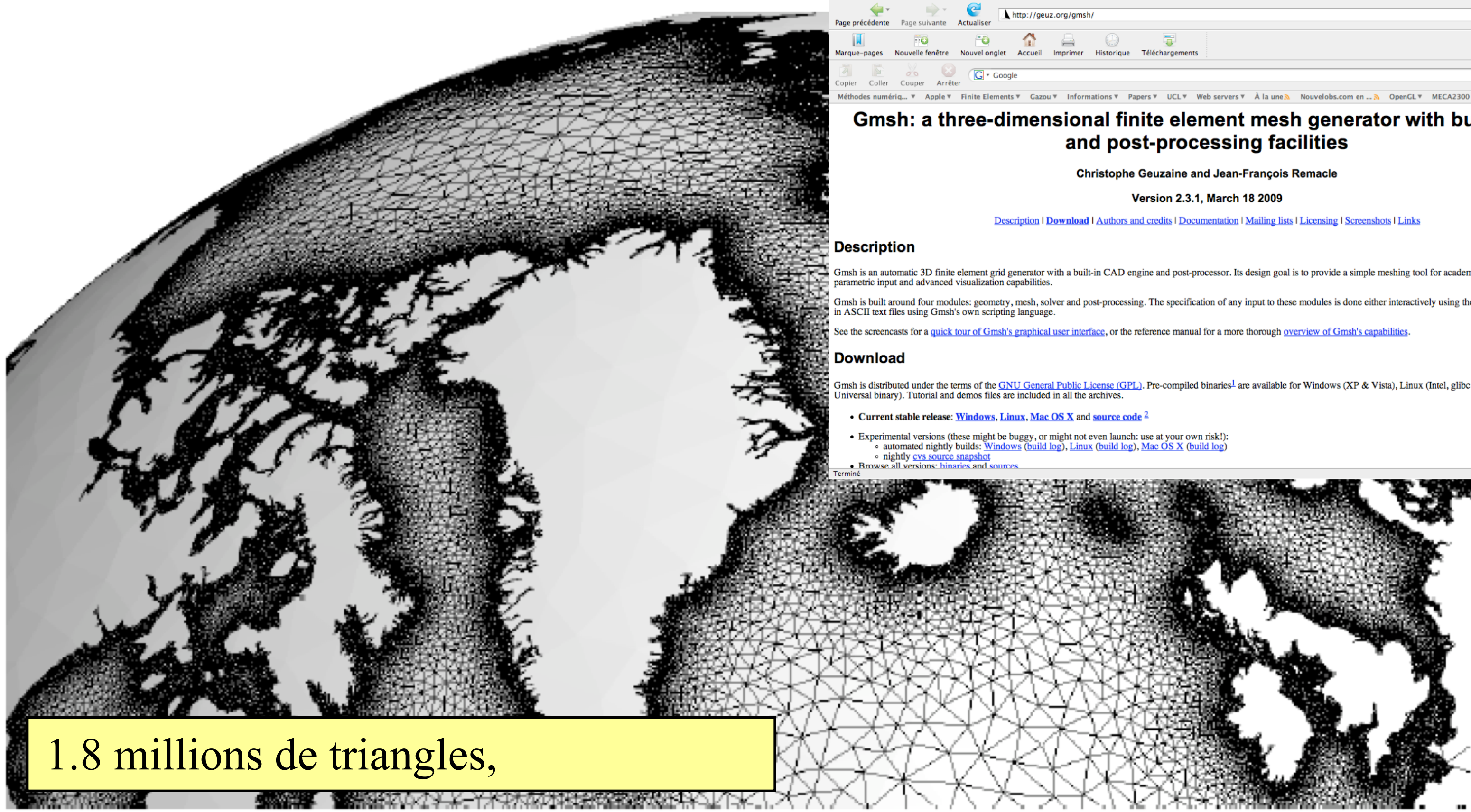
Les méthodes numériques
A quoi cela sert-il ?



Second-generation Louvain-la-Neuve Ice-ocean Model





Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities

Christophe Geuzaine and Jean-François Remacle

Version 2.3.1, March 18 2009

Description | Download | Authors and credits | Documentation | Mailing lists | Licensing | Screenshots | Links

Description

Gmsh is an automatic 3D finite element grid generator with a built-in CAD engine and post-processor. Its design goal is to provide a simple meshing tool for academic problems with parametric input and advanced visualization capabilities.

Gmsh is built around four modules: geometry, mesh, solver and post-processing. The specification of any input to these modules is done either interactively using the graphical user interface or in ASCII text files using Gmsh's own scripting language.

See the screencasts for a quick tour of Gmsh's graphical user interface, or the reference manual for a more thorough overview of Gmsh's capabilities.

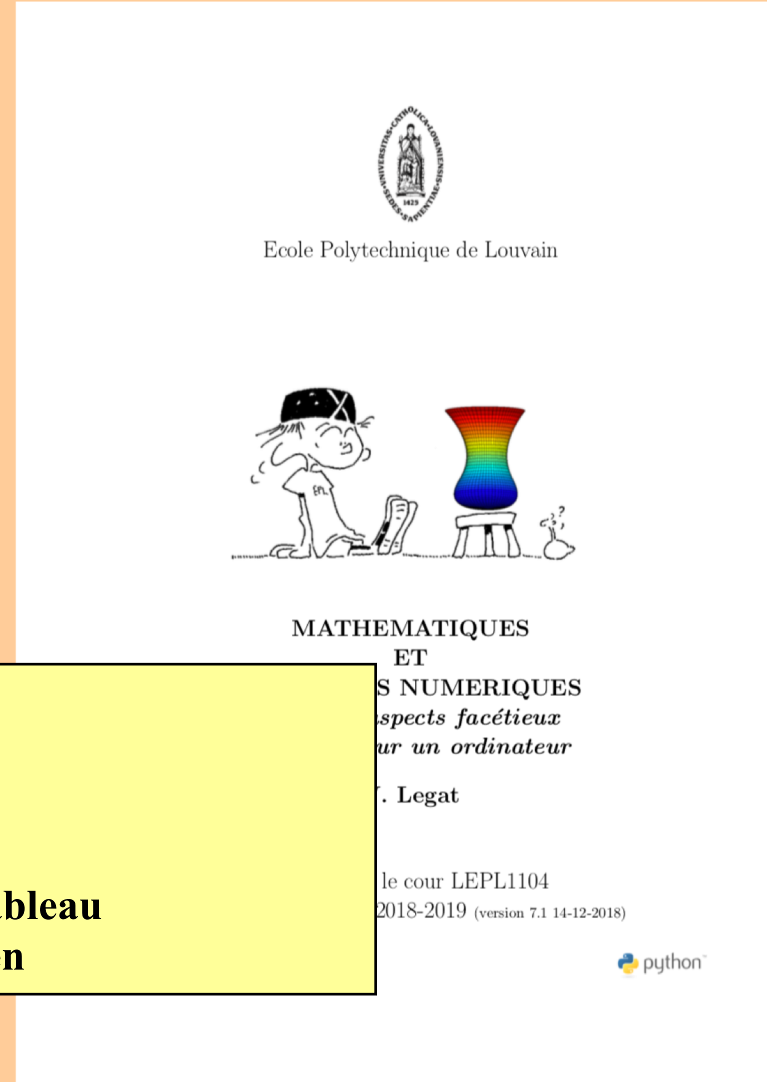
Download

Gmsh is distributed under the terms of the GNU General Public License (GPL). Pre-compiled binaries[1] are available for Windows (XP & Vista), Linux (Intel, glibc 2.3) and Mac OS X (10.5, Universal binary). Tutorial and demos files are included in all the archives.

- **Current stable release**: **Windows**, **Linux**, **Mac OS X** and **source code** [2]
- Experimental versions (these might be buggy, or might not even launch: use at your own risk!):
  - automated nightly builds: Windows (build log), Linux (build log), Mac OS X (build log)
  - nightly cvs source snapshot

1.8 millions de triangles,

Notes de cours



Notes de cours
Enoncés des exercices
Solutions des exercices
Transparents du cours
Ce qui est noté sur le tableau
= la matière de l'examen

disponibles au SICI et on-line sur le Web

Plan des cours de méthodes numériques

Comment interpoler une fonction ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux valeurs initiales ?

Comment résoudre numériquement un problème aux conditions frontières ?

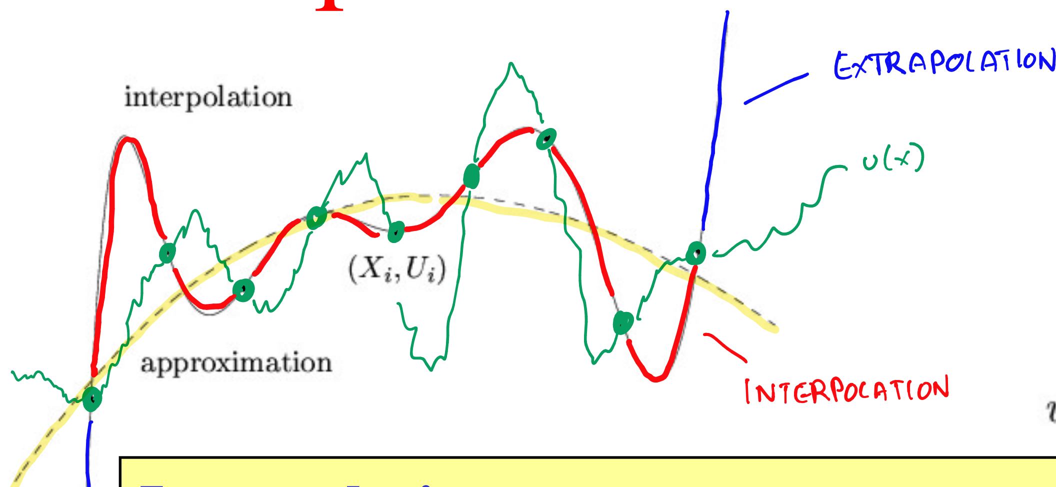
Et les équations non linéaires ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation différentielle ?

Interpolation, approximation et extrapolation…



Fonctions de base spécifiées a priori

$$u(x) \approx \overbrace{u^h(x)}^{\text{INTERPOLATION}} = \sum_{j=0}^{n} a_j \ \phi_j(x)$$

Paramètres inconnus

Interpolation :

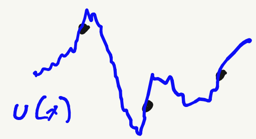
la fonction $u^h(x)$ passe exactement par les points. Valeurs interpolées entre les points et valeurs extrapolées hors de l'intervalle.

Approximation :

la fonction $u^h(x)$ ne passe pas par les points, mais s'en rapproche selon un critère à définir

$\boxed{1}$

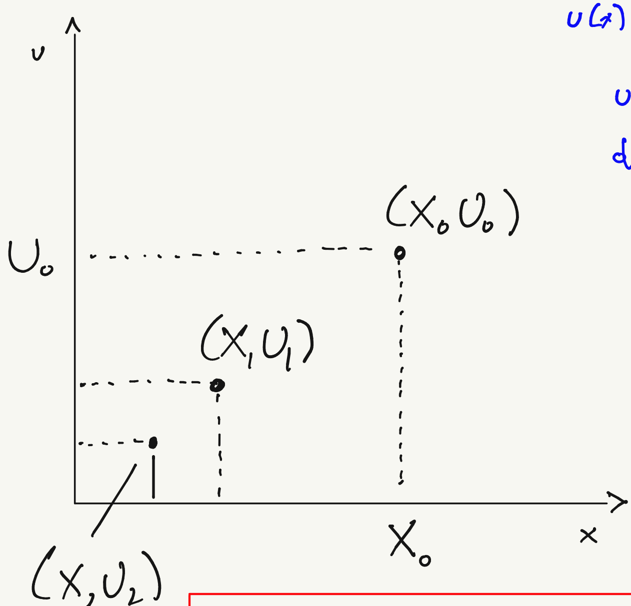
MATRICE DE VANDERMONDE



$u^h(x) = a + bx + cx^2$

$u(x) \in \mathcal{U}$ $\quad$ $u^h(x) \in \mathcal{U}^h$

$\dim(\mathcal{U}) = \infty$ $\quad$ $\dim(\mathcal{U}^h) = 3$



$$u(x) \simeq u^h(x) = \underbrace{a_0}_{\in \mathbb{R}} \underbrace{\varphi_0(x)}_{\in \mathcal{U}^h} + a_1 \varphi_1(x) + a_2 \varphi_2(x)$$

INCONNUS $\quad$ CONNUES A PRIORI

? $a_0 \quad a_1 \quad a_2$ tels que

$$\underbrace{u(X_0)}_{U_0} = u^h(X_0)$$

$$U_1 = u^h(X_1)$$

$$U_2 = u^h(X_2)$$

EXEMPLE

$\phi_0(x) = 1$

$\phi_1(x) = x$

$\phi_2(x) = x^2$

POLYNOME DE DEGRE 2 PASSANT PAR 3 POINTS

$X_0 \; X_1 \; X_2$

? $a_0 \quad a_1 \quad a_2$ tels que

$$\underbrace{u(X_0)}_{U_0} = u^h(X_0)$$

$$U_1 = u^h(X_1)$$

$$U_2 = u^h(X_2)$$

$$\begin{cases} a_0\varphi_0(X_0) + a_1\varphi_1(X_0) + a_2\varphi_2(X_0) = U_0 \\ a_0\varphi_0(X_1) + a_1\varphi_1(X_1) + a_2\varphi_2(X_1) = U_1 \\ a_0\varphi_0(X_2) + a_1\varphi_1(X_2) + a_2\varphi_2(X_2) = U_2 \end{cases}$$

$$\underbrace{\begin{bmatrix} \varphi_0(X_0) & \varphi_1(X_0) & \varphi_2(X_0) \\ & \varphi_1(X_1) & \varphi_2(X_1) \\ & & \varphi_2(X_2) \end{bmatrix}}_{\underline{\underline{A}}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}}_{\underset{\sim}{x}} = \underbrace{\begin{bmatrix} U_0 \\ U_1 \\ U_2 \end{bmatrix}}_{\underset{\sim}{u}}$$

MATRICES ET LES VECTEURS SONT EN GRAS !

EXEMPLE

$$\phi_0(x) = 1$$
$$\phi_1(x) = x$$
$$\phi_2(x) = x^2$$

POLYNOME DE DEGRE 2
PASSANT PAR 3 POINTS
$X_0 \; X_1 \; X_2$

$$\underline{\underline{A}} \cdot \underset{\sim}{x} = \underset{\sim}{b}$$

FORGET ALGEBRA !

$$\cancel{\underset{\sim}{x} = \underline{\underline{A}}^{-1} \cdot \underset{\sim}{b}}$$

ON RESOUT UN SYSTEME

ON NE L'INVERSE PAS !

SYLLABUS ? a_i

$$\sum_{i=1}^{n} a_i \varphi_i(X_j) = U_j$$

$$j = 0 \ldots n$$

Problème de l'interpolation

Trouver $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\underbrace{\sum_{j=0}^{n} a_j \, \phi_j(X_i)}_{u^h(X_i)} = U_i \qquad\qquad i = 0, 1, \ldots, n.$$

$$\begin{bmatrix} \phi_0(X_0) & \phi_1(X_0) & \ldots & \phi_n(X_0) \\ \phi_0(X_1) & \phi_1(X_1) & \ldots & \phi_n(X_1) \\ \phi_0(X_2) & \phi_1(X_2) & \ldots & \phi_n(X_2) \\ \phi_0(X_3) & \phi_1(X_3) & \ldots & \phi_n(X_3) \\ \phi_0(X_4) & \phi_1(X_4) & \ldots & \phi_n(X_4) \\ \vdots & \vdots & & \vdots \\ \phi_0(X_n) & \phi_1(X_n) & \ldots & \phi_n(X_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ \vdots \\ U_n \end{bmatrix}$$

Cas particulier :
Interpolation polynomiale

$$\phi_j(x) = x^j \qquad j = 0, 1, 2, \ldots, n.$$

$$\boxed{u^h(x) = \sum_{j=0}^{n} a_j \; x^j}$$

Matrice de Vandermonde

$$\begin{bmatrix} 1 & X_0 & \ldots & X_0^n \\ 1 & X_1 & \ldots & X_1^n \\ \vdots & \vdots & & \vdots \\ 1 & X_n & \ldots & X_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_n \end{bmatrix}$$

Unicité de l'interpolation polynomiale :

Il existe un et un seul polynôme d'interpolation de degré n au plus qui passe par n + 1 points d'abscisses distinctes.

Alexandre-Théophile Vandermonde

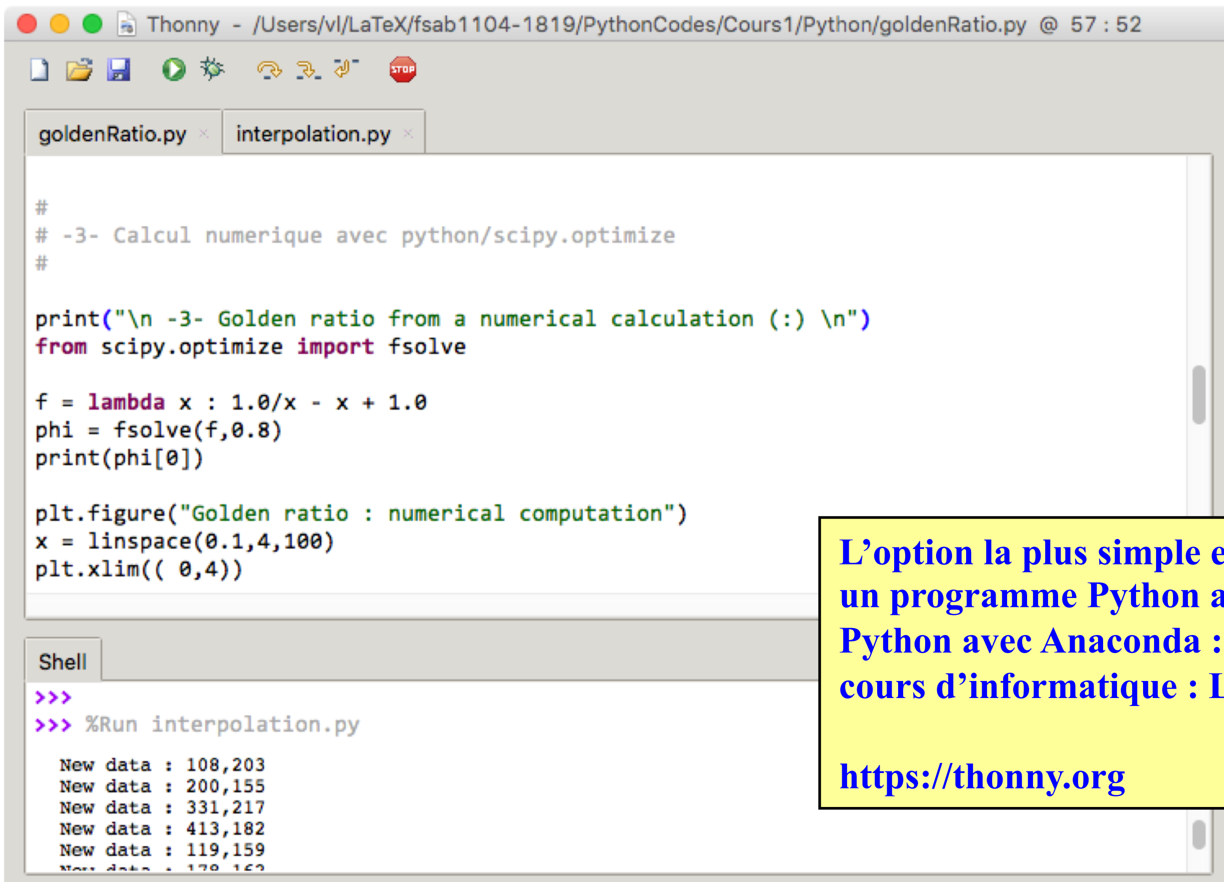
Né le 28 février 1735 à Paris
Décédé le 1er janvier 1796 à Paris

Perhaps the name of Vandermonde is best known today for the Vandermonde determinant. While it is certainly true that he made a major contribution to the theory of determinants, yet nowhere in his four mathematical papers does this determinant appear. It is rather strange, therefore, that this determinant should be named after him and several authors have puzzled over the fact for some time.

Lesbesgue's conjecture (first published in 1940) that it resulted for someone misreading Vandermonde's notation, and therefore believing that this determinant was in his work, seems the most likely.

http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Vandermonde.html

Un éditeur Python pour débutants !



L'option la plus simple et la facile pour exécuter et éditer un programme Python après avoir installé l'interpréteur Python avec Anaconda : c'est aussi l'option choisie dans le cours d'informatique : LEPL1401

https://thonny.org

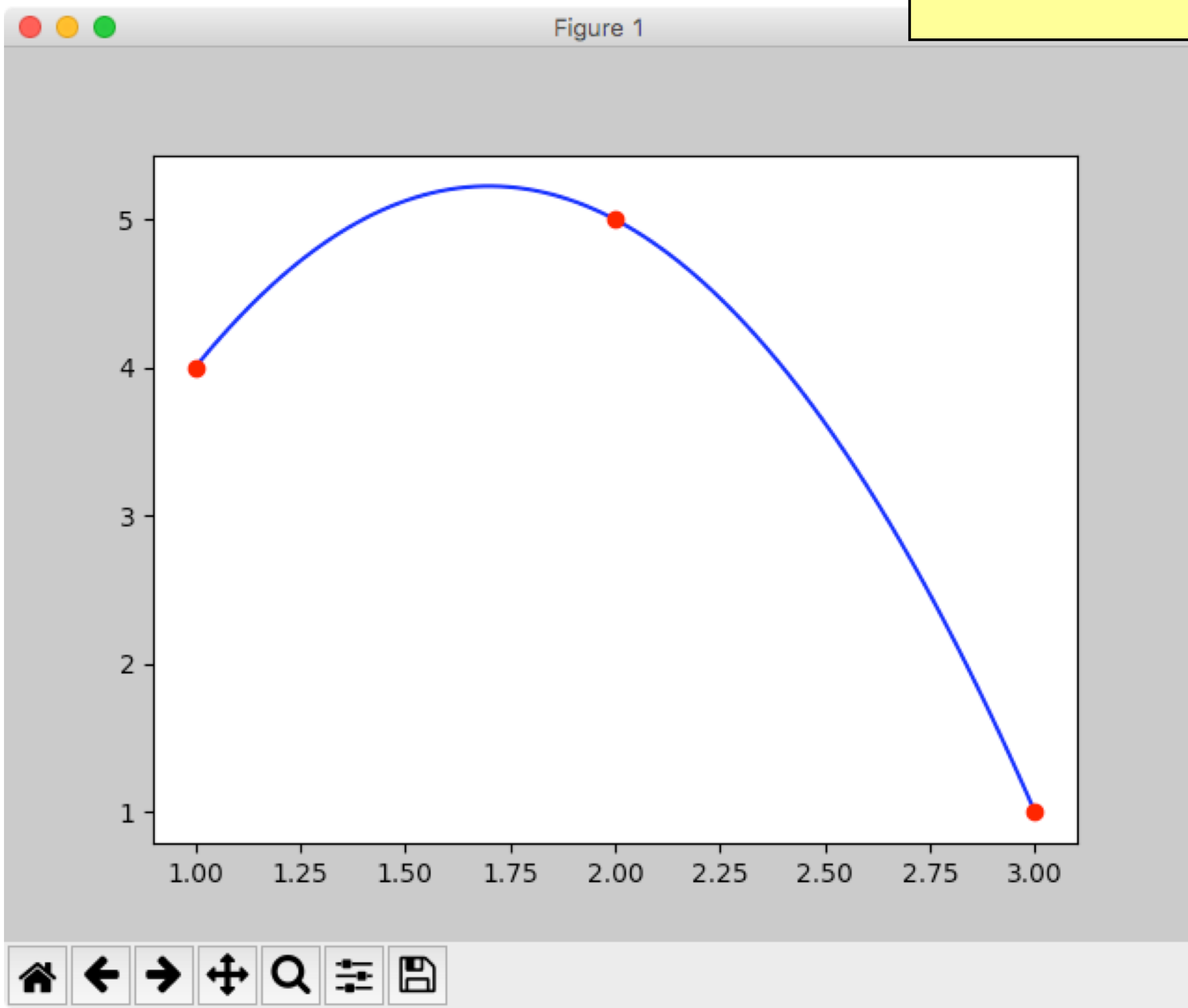
Comment avoir beaucoup de Python sur votre ordinateur ?

C'est gratuit et open source !
C'est le langage pour apprendre l'informatique dans le cours LEPL1401 dès cette année !

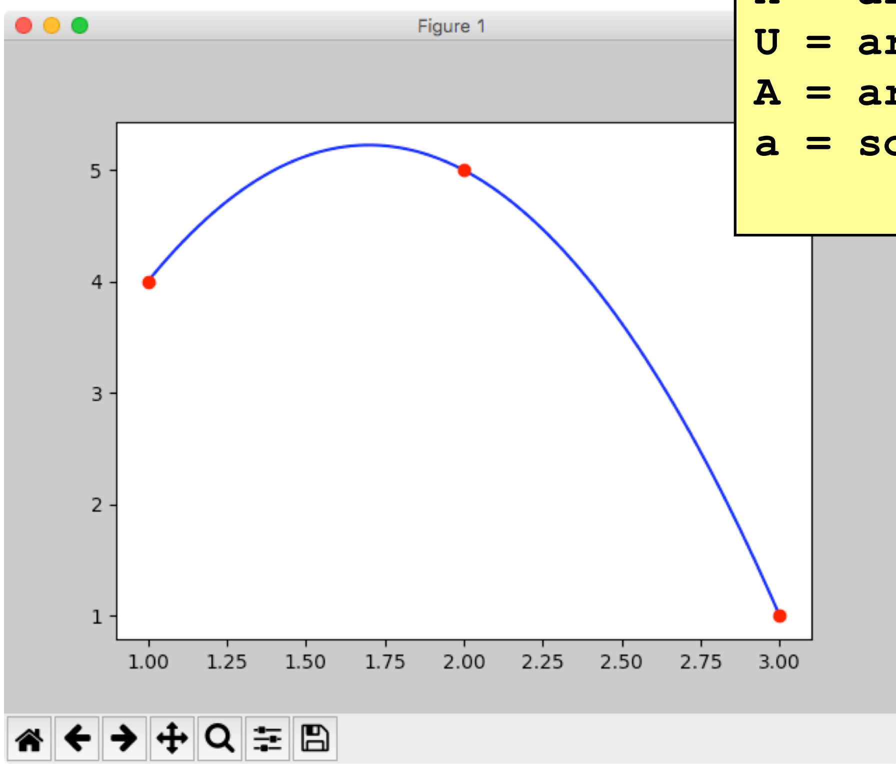
https://www.anaconda.com/

Faisons le calcul

```
X = array([1,2,3])
U = array([4,5,1])
A = array([X**0,X**1,X**2]).T
a = solve(A,U)
```



Faisons le calcul avec numpy




```
from numpy import *
from numpy.linalg import solve

X = array([1,2,3])
U = array([4,5,1])
A = array([X**0,X**1,X**2]).T
a = solve(A,U)
```

A propos de la résolution d'un système linéaire...

```
A = inv(A)
x = A @ b
```

```
x = solve(A,b)
```

On résout un système linéaire,
on ne l'inverse jamais….
(J. Meinguet)

A frequent misuse of **`inv`** arises when solving the system of linear equations. One way to solve this is with

```
x = inv(A)*b
```

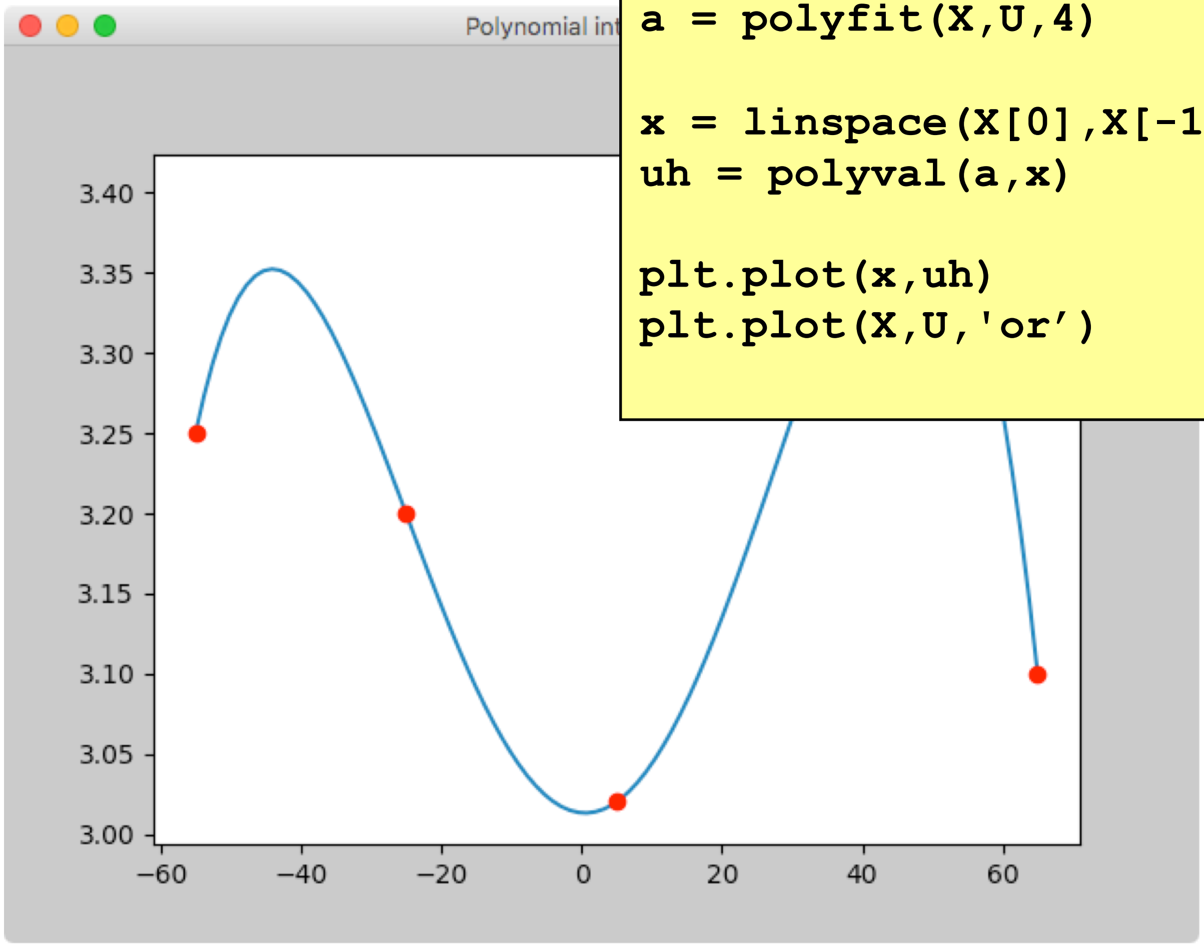
A better way, from both an execution time and numerical accuracy standpoint, is to use the matrix division operator

```
x = A\b
```

This produces the solution using Gaussian elimination, without forming the inverse.

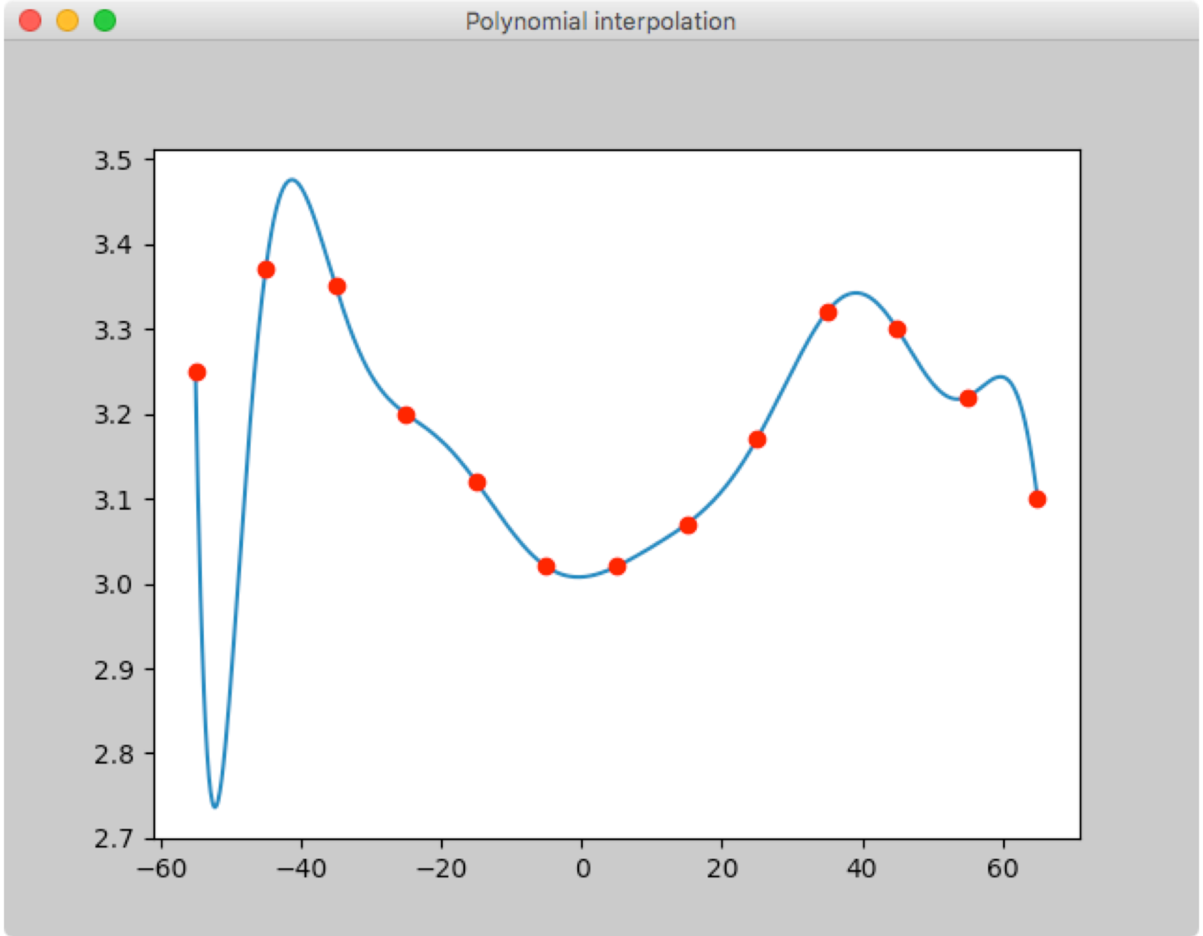
Exemple

```
from numpy import *
from matplotlib import pyplot as plt

X = [ -55, -25,   5,  35,  65]
U = [3.25,3.20,3.02,3.32,3.10]
a = polyfit(X,U,4)

x = linspace(X[0],X[-1],100)
uh = polyval(a,x)

plt.plot(x,uh)
plt.plot(X,U,'or')
```

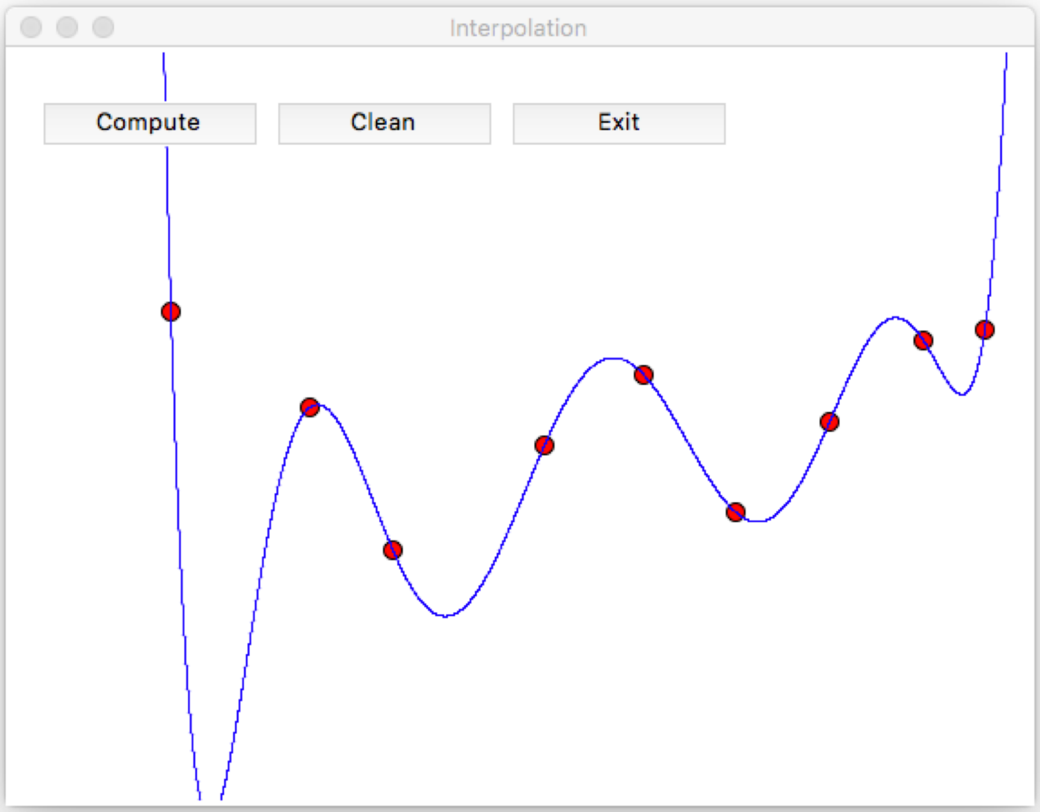


Davantage de données…

Latitude	
65	3.10
55	3.22
45	3.30
35	3.32
25	3.17
15	3.07
5	3.02
-5	3.02
-15	3.12
-25	3.20
-35	3.35
-45	3.37
-55	3.25



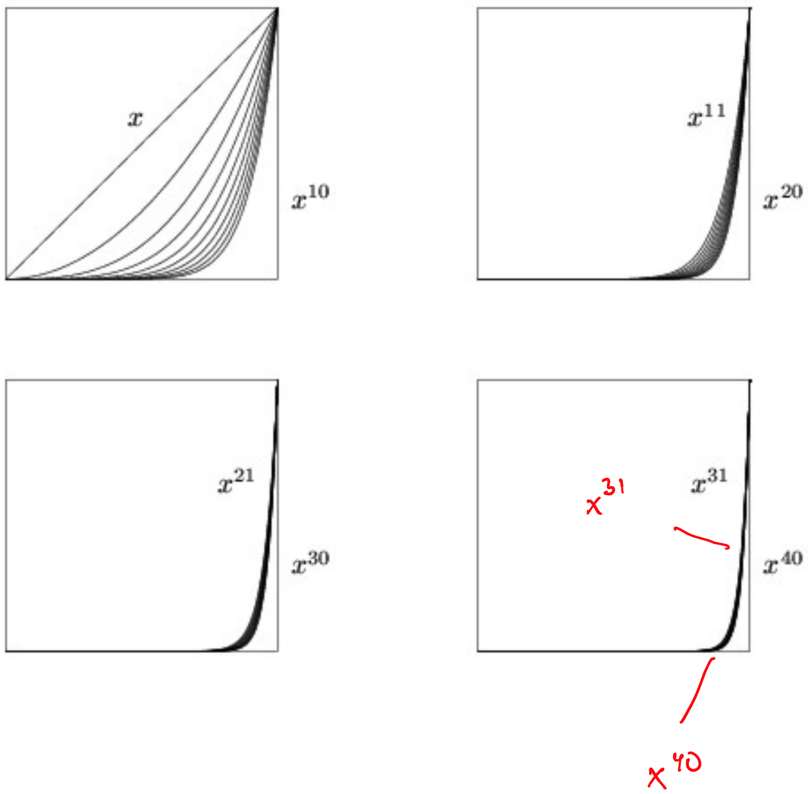
Le polynôme d'interpolation est un être bien turbulent…



Les monômes

$$u^h(x) = \sum_{j=0}^{n} a_j \boxed{x^j}$$

$$\begin{bmatrix} 1 & X_0 & \dots & X_0^n \\ 1 & X_1 & \dots & X_1^n \\ \vdots & \vdots & & \vdots \\ 1 & X_n & \dots & X_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_n \end{bmatrix}$$



Matrice de Vandermonde…

Le système linéaire devient de plus en plus mal conditionné, lorsque n augmente !

C'est quoi un système mal conditionné ?

Données

$$\begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Solution exacte :
a=2.0 et b=0.0

Système linéaire mal conditionné

Un système linéaire est mal conditionné si une petite variation des données entraîne une très grande variation des résultats.

Il s'agit d'une propriété qui est directement liée au système linéaire et qui est donc totalement indépendante de la méthode numérique pour résoudre ce système.

Perturbons un peu les deux systèmes

Système mal conditionné

Solution perturbée :
a=1.0 et b=1.0

Données légèrement perturbées

$$\begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \begin{bmatrix} a_\epsilon \\ b_\epsilon \end{bmatrix} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} a_\epsilon \\ b_\epsilon \end{bmatrix} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix}$$

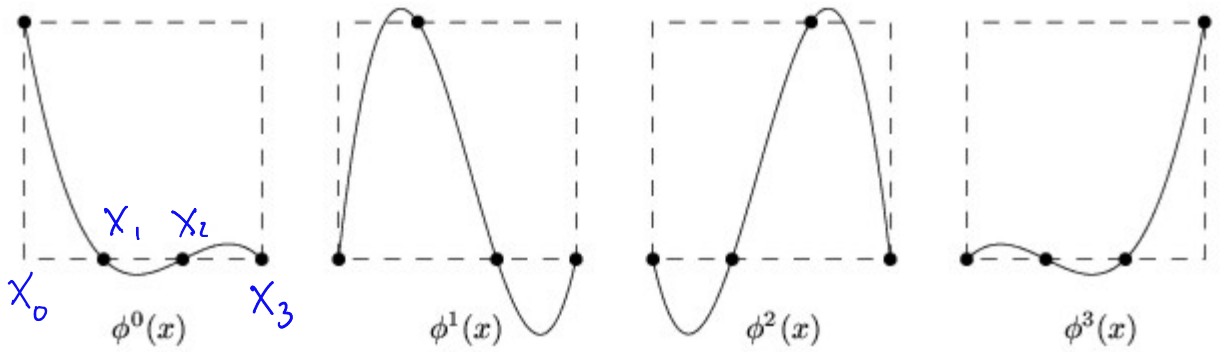
Solution perturbée :
a=1.99999 et b=0.00001

Système bien conditionné

Comment savoir si un système est bien conditionné ?

```
======== Good condition
   x[0] =  2.0000000e+00 and xpert[0] =  1.9999900e+00
   x[1] =  0.0000000e+00 and xpert[1] =  1.0000000e-05
   condition number        =     1.232e+01
   determinant             =     1.000e+01
   lambda[0]               =     9.010e-01
   lambda[1]               =     1.110e+01
   lambda[0] * lambda[1] =     1.000e+01
   lambda[1] / lambda[0] =     1.232e+01
======== Bad condition
   x[0] =  2.0000000e+00 and xpert[0] =  1.0000000e+00
   x[1] =  0.0000000e+00 and xpert[1] =  1.0000000e+00
   condition number        =     4.000e+04
   determinant             =     1.000e-04
   lambda[0]               =     5.000e-05
   lambda[1]               =     2.000e+00
   lambda[0] * lambda[1] =     1.000e-04
   lambda[1] / lambda[0] =     4.000e+04
bash-3.2$
```

```
A = [[1,1],[1,1.0001]]
b = [2,2]
bpert = [2,2.0001]

x     = solve(A,b)
xpert = solve(A,bpert)

for i in range(len(x)):
  print("   x[%d] = %14.7e and xpert[%d] = %14.7e" % (i,x[i],i,xpert[i]))

lambdas,vect = eig(A)
print("   condition number       = %12.3e" % cond(A))
print("   determinant            = %12.3e" % det(A))
print("   lambda[0]              = %12.3e" % lambdas[0])
print("   lambda[1]              = %12.3e" % lambdas[1])
print("   lambda[0] * lambda[1] = %12.3e" % (lambdas[0]*lambdas[1]))
print("   lambda[1] / lambda[0] = %12.3e" % (lambdas[1]/lambdas[0]))
```



$$\phi_i(x) \;=\; \frac{(x-X_0)(x-X_1)\;\ldots\;(x-X_{i-1})(x-X_{i+1})\;\ldots\;(x-X_n)}{(X_i-X_0)(X_i-X_1)\ldots(X_i-X_{i-1})(X_i-X_{i+1})\ldots(X_i-X_n)}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \\ 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \end{bmatrix}$$

Les fonctions de base de Lagrange

Idée…

$$u^h(x) = \sum_{i=0}^{n} U_i \phi_i(x)$$

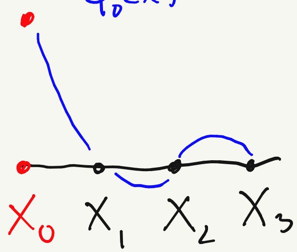
Choisir les fonctions de base afin que la matrice du système soit la matrice unité !

Même solution par unicité de l'interpolation !

Pas de problème de conditionnement !

$\boxed{2}$ POLYNOMES DE LAGRANGE

$$\phi_0(x) = a + bx + cx^2 + dx^3$$

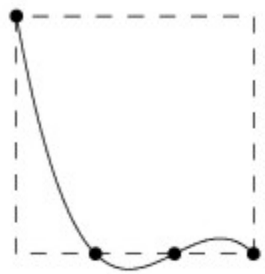


4 INCONNUES

4 CONDITIONS

$$\begin{cases} \phi_0(X_0) = 1 \\ \phi_0(X_1) = 0 \\ \phi_0(X_2) = 0 \\ \phi_0(X_3) = 0 \end{cases}$$

$$\phi_0(x) = \frac{(x - X_1)(x - X_2)(x - X_3)}{(X_0 - X_1)(X_0 - X_2)(X_0 - X_3)}$$



$\phi^0(x)$

Les fonctions de base de Lagrange

PAS DE TERME $(x - X_1)$

$$\phi_i(x) \quad = \quad \frac{(x - X_0)(x - X_1)\ \ldots\ (x - X_{i-1})(x - X_{i+1})\ \ldots\ (x - X_n)}{(X_i - X_0)(X_i - X_1)\ldots(X_i - X_{i-1})(X_i - X_{i+1})\ldots(X_i - X_n)}$$

Erreur d'interpolation

Fonction inconnue

$$e^h(x) = u(x) - u^h(x)$$

Interpolation polynomiale

Théorème 1.1. *Soit $X_0 < X_1 < \ldots < X_n$ les abscisses des points d'interpolation. Si la fonction $u(x)$ est définie sur l'intervalle $[X_0, X_n]$ et qu'elle est $(n+1)$ fois dérivable sur $]X_0, X_n[$, alors pour tout $x \in]X_0, X_n[$, il existe $\xi(x) \in]X_0, X_n[$ tel que :*

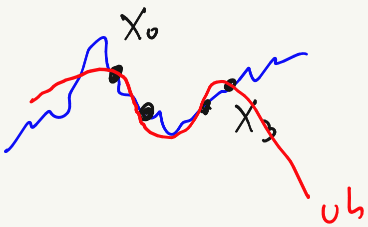
$$e^h(x) = \frac{u^{(n+1)}(\xi(x))}{(n+1)!}\,(x - X_0)(x - X_1)(x - X_2)\cdots(x - X_n).$$

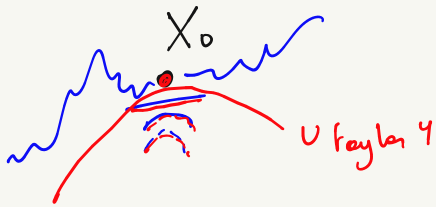
3 **ERREUR D'INTERPOLATION**

INTERPOLATION

$n = 4$

$$u - u^h(x) = u''''(\xi(x)) \frac{(x - X_0)(x - X_1)(x - X_2)(x - X_3)}{4!}$$



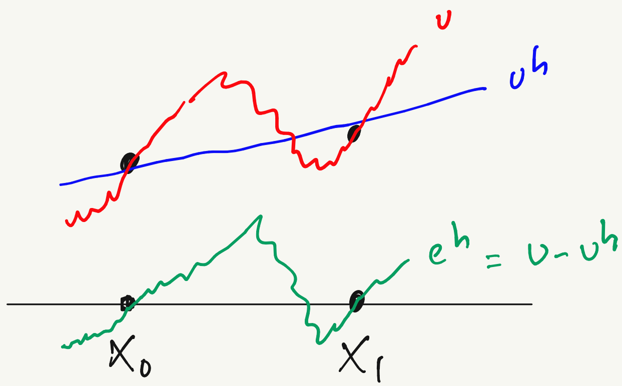


TAYLOR

$$u(x) - u^{taylor\,4}(x) = u''''(\xi(x)) \frac{(x - X_0)^4}{4!}$$

DEMONSTRATION :-)

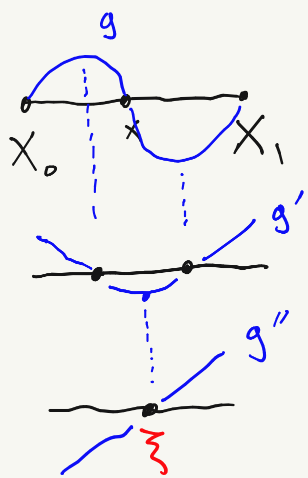
$n = 1$



$$g(t) = \underbrace{u(t) - u^h(t)}_{e^h(t)} - e^h(x)\,\underbrace{\frac{(t-X_0)(t-X_1)}{(x-X_0)(x-X_1)}}$$

$t = X_0$	$= 0$
$t = X_1$	$= 0$
$t = x$	$= e^h(x)$

$t = X_0$	$= 0$
$t = X_1$	$= 0$
$t = x$	$= e^h(x)$



$\exists\, \xi(x)$ tel que

$$g''(\xi) = 0$$

$$u''(\xi) - \frac{e^h(x)\,2}{(x-X_0)(x-X_1)} = 0$$

$$e^h(x) = u''(\xi(x))\,\frac{(x-X_0)(x-X_1)}{2!}$$

$\square$

$$e^h(x) = \frac{u''(\xi)\,(x-X_0)(x-X_1)}{2!}$$

$$|e^h(x)| \leqslant \underbrace{\left|\frac{u''(\xi(x))}{2!}\right|}_{\leqslant \frac{C_2}{2}} \, \underbrace{\left|(x-X_0)(x-X_1)\right|}_{f(x)\, \leqslant\, \frac{h^2}{4}}$$

$$\underbrace{|e^h(x)|}_{} \leqslant \underbrace{\frac{C_2}{2}\,\frac{h^2}{4}}_{O(h^2)}$$

"$e_x(h)$"

VU COMME UNE FONCTION DE h
POUR UN x FIXE

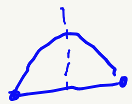
$$\exists\, C \text{ tel que } \left|\frac{f(h)}{h^2}\right| \leqslant C$$

AU VOISINAGE DE 0 :-)

$X_0 = 0$ $\quad$ $X_1 = h$

h

$$f(x) = x(x-h)$$
$$f'(x) = 2x - h$$



x_m

$$\underbrace{f'(x_m)}_{=0} = 2x_m - h$$

$$x_m = \frac{h}{2}$$

$$f(x_m) = -\frac{h^2}{4}$$

$$\underbrace{|e^h(x)|} \leq \underbrace{\frac{C_2}{2}\,\frac{h^2}{4}}_{O(h^2)}$$

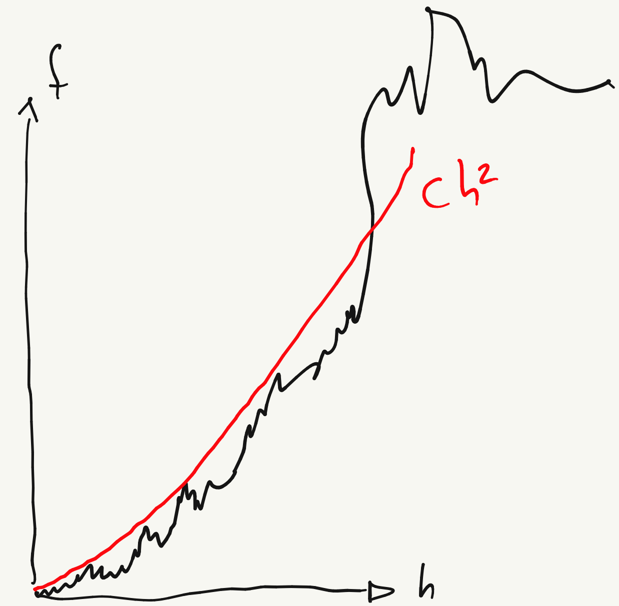
2 = ORDRE DE L'ERREUR

"$e_x(h)$"

VU COMME UNE FONCTION DE h
POUR UN x FIXE

$$\exists\, C \text{ tel que } \left|\frac{f(h)}{h^2}\right| \leq C$$

AU VOISINAGE DE 0 :-)



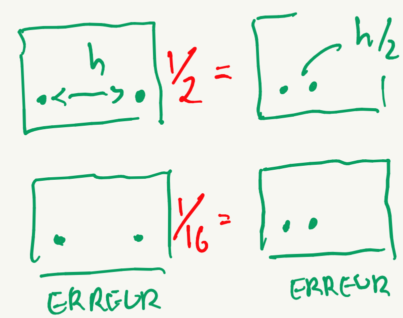
ORDRE
ASYMPTOTIQUE

$m = 2$

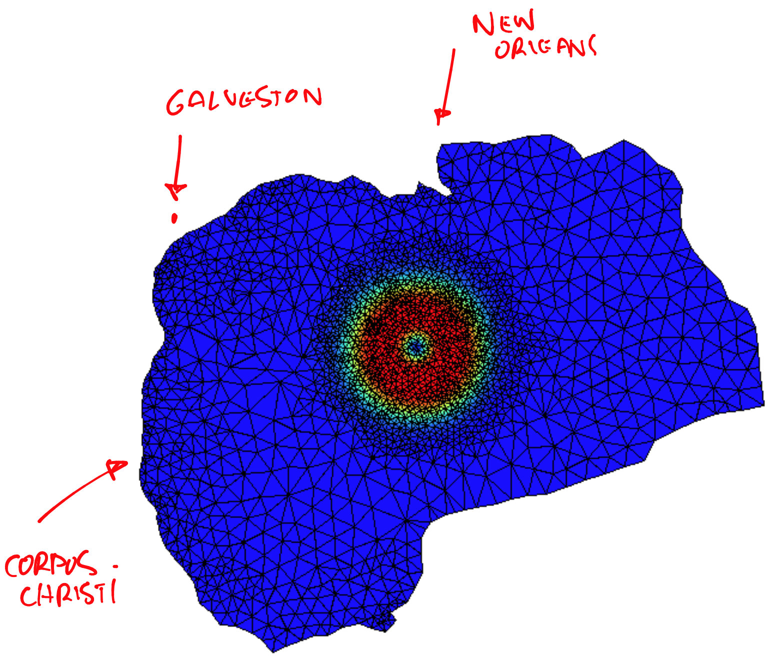
$$\underbrace{\boxed{C\left[\frac{h}{2}\right]^m}}_{\text{ERREUR POUR } h/2} = \frac{1}{2^m}\underbrace{\boxed{C\,h^m}}_{\text{ERREUR POUR } h}$$

$m = 1$

$m = 4$



Est-ce utile ?



Il vaut mieux mettre beaucoup de points à l'endroit où on souhaite que la précision soit la plus grande possible...

Intuitivement, on s'en serait un peu douté !

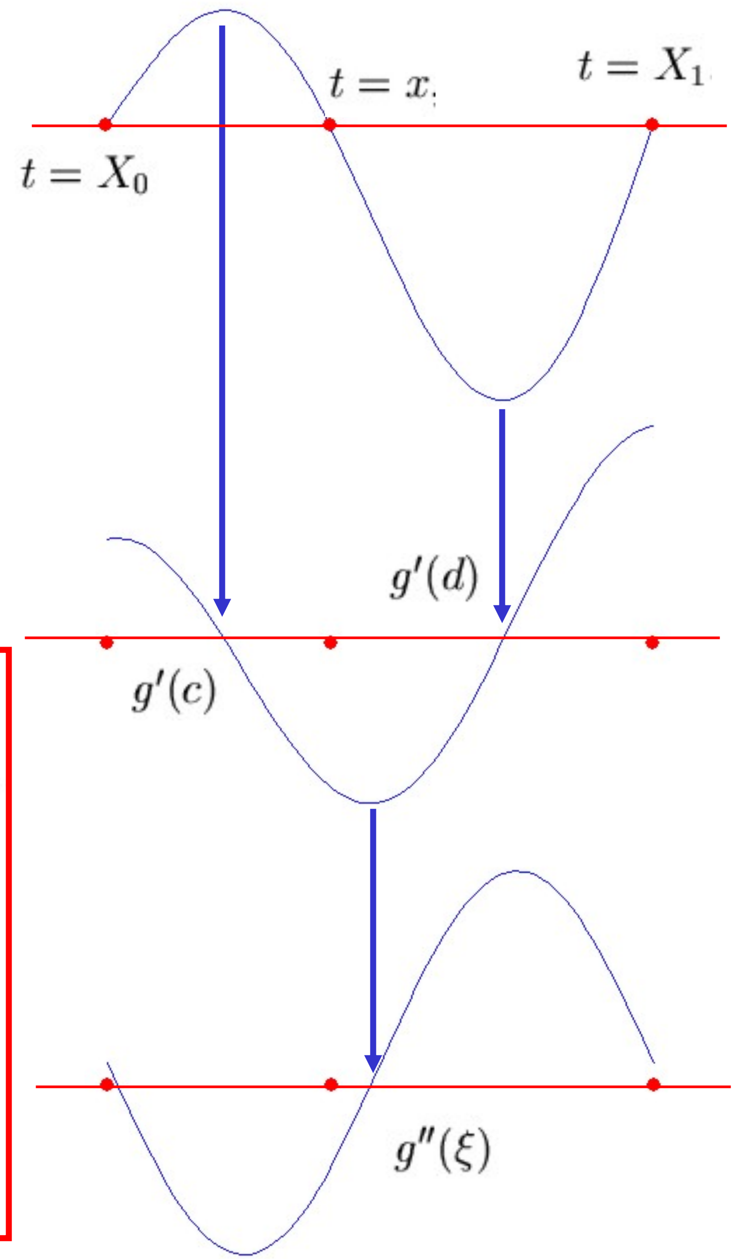
Paul-Emile Bernard et al.(2006)
http://www.climate.be/slim

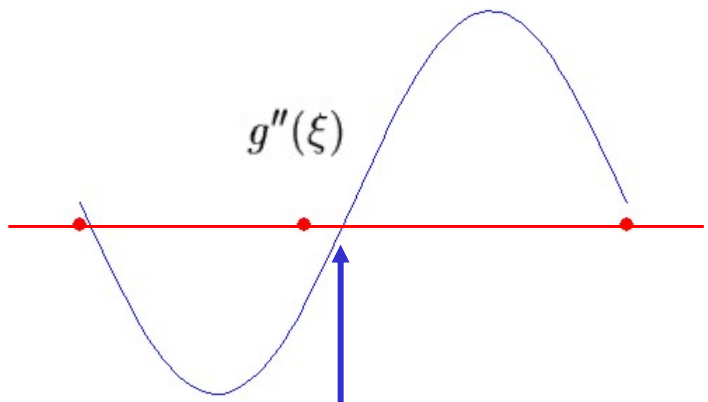
Nous allons démontrer uniquement le cas n=1
Ceci est le début de la démonstration :-)

Utilisons le théorème de Rolle…

$$g(t) = u(t) - u^h(t) - e^h(x)\frac{(t - X_0)(t - X_1)}{(x - X_0)(x - X_1)}$$

On a choisi cette fonction car elle s'annule en t= x , t=X_0 et t = X_1





$$
\begin{aligned}
g''(\xi) &= 0 \\
&\downarrow \\
u''(\xi) - \underbrace{(u^h)''(\xi)}_{=0} - e^h(x)\frac{2}{(x-X_0)(x-X_1)} &= 0 \\
&\downarrow \\
e^h(x) &= \frac{u''(\xi)}{2}(x-X_0)(x-X_1) \qquad \square
\end{aligned}
$$

Ceci est la fin de la démonstration :-)

$$|e^h(x)| \leq \frac{C_{n+1}}{(n+1)!} \underbrace{|(x-X_0)(x-X_1)\cdots(x-X_n)|}_{\leq n!\, h^{n+1}/4}$$

$\downarrow$ En définissant une nouvelle constante $C = \frac{n! C_{n+1}}{4(n+1)!}$

$$|e^h(x)| \leq C h^{n+1}$$

Borne d'erreur

$\downarrow$

$$e^h(x) = \mathcal{O}(h^{n+1})$$

Définition 1.1.

On dit qu'une fonction $f(h)$ est un grand ordre de h^m au voisinage de 0, s'il existe une constante C telle que :

$$\left|\frac{f(h)}{h^m}\right| \leq C$$

au voisinage de 0. On écrit alors $f(h) = \mathcal{O}(h^m)$.

Ordre d'une méthode numérique

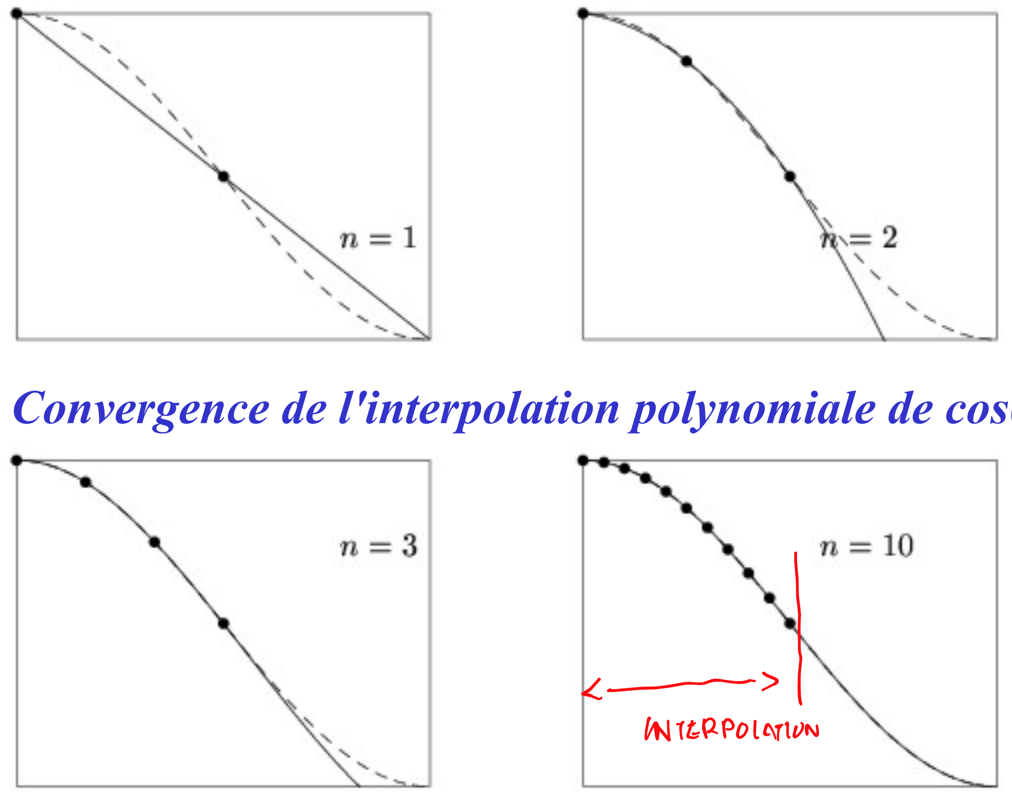
Définition 1.2. *Une approximation numérique dont le terme d'erreur est $\mathcal{O}(h^m)$ est dite d'ordre m.*

Polynôme de Taylor de degré n est souvent d'ordre n+1
Interpolation polynomiale par n+1 points équidistants est souvent d'ordre n+1

Signification intuitive de l'ordre d'une méthode numérique

$$C\left(\frac{h}{2}\right)^m = \frac{1}{2^m}\, Ch^m$$

Convergence



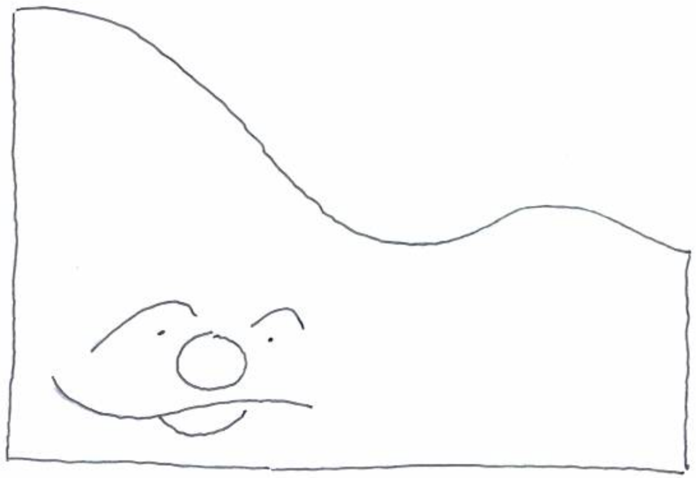
Convergence de l'interpolation polynomiale de cos(x)

$$e^h(x) = u(x) - u^h(x)$$

Définition 1.3. *Une interpolation est dite* convergente *si l'erreur d'interpolation tend vers zéro lorsque le nombre de degrés de liberté, c'est-à-dire n tend vers l'infini :*

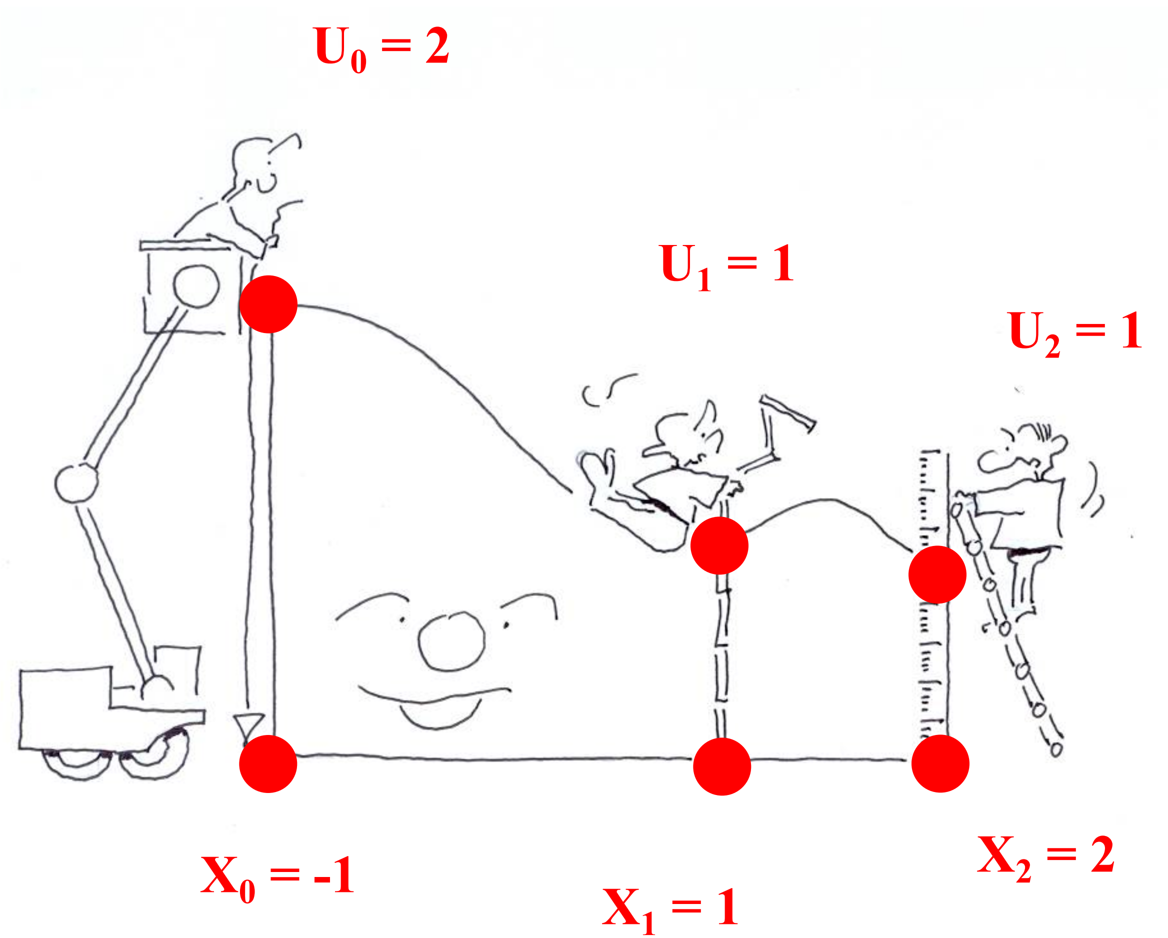
$$\lim_{n \to \infty} e^h(x) = 0 \quad \textit{pour } x \in [X_0, X_n].$$

Une courbe….

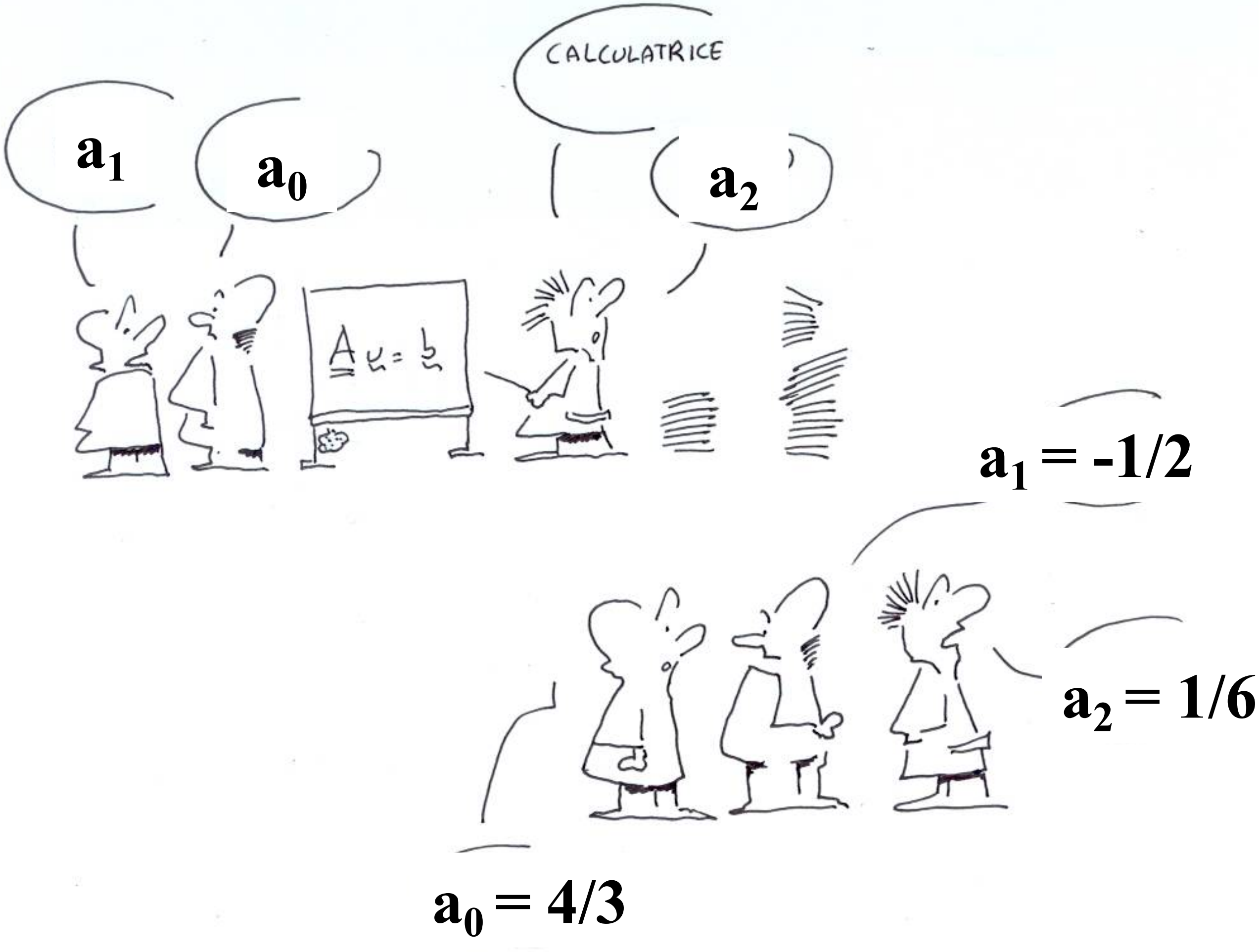


Comment la représenter sur un ordinateur ?

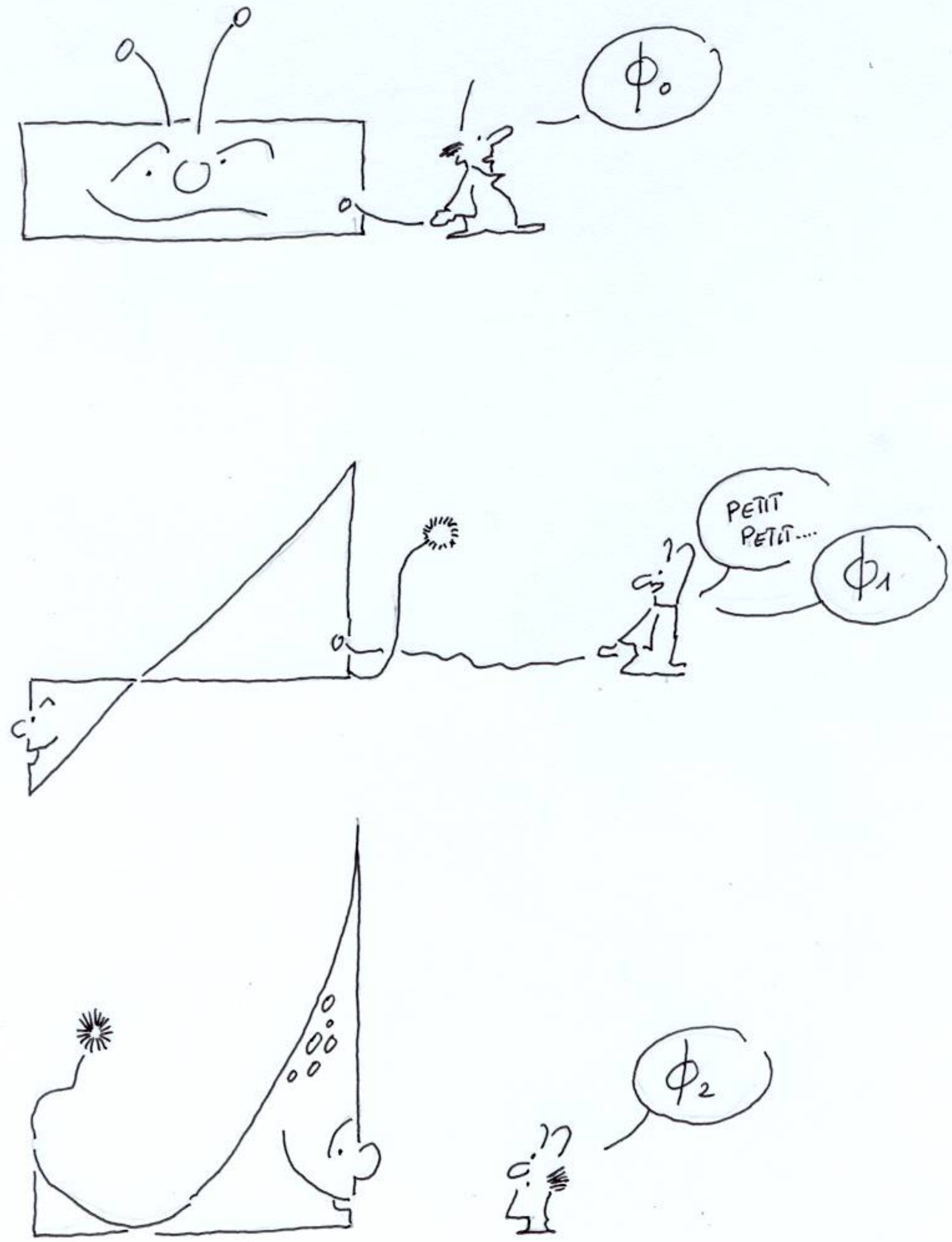
Prise de mesures



Utilisation des ressources informatiques facultaires



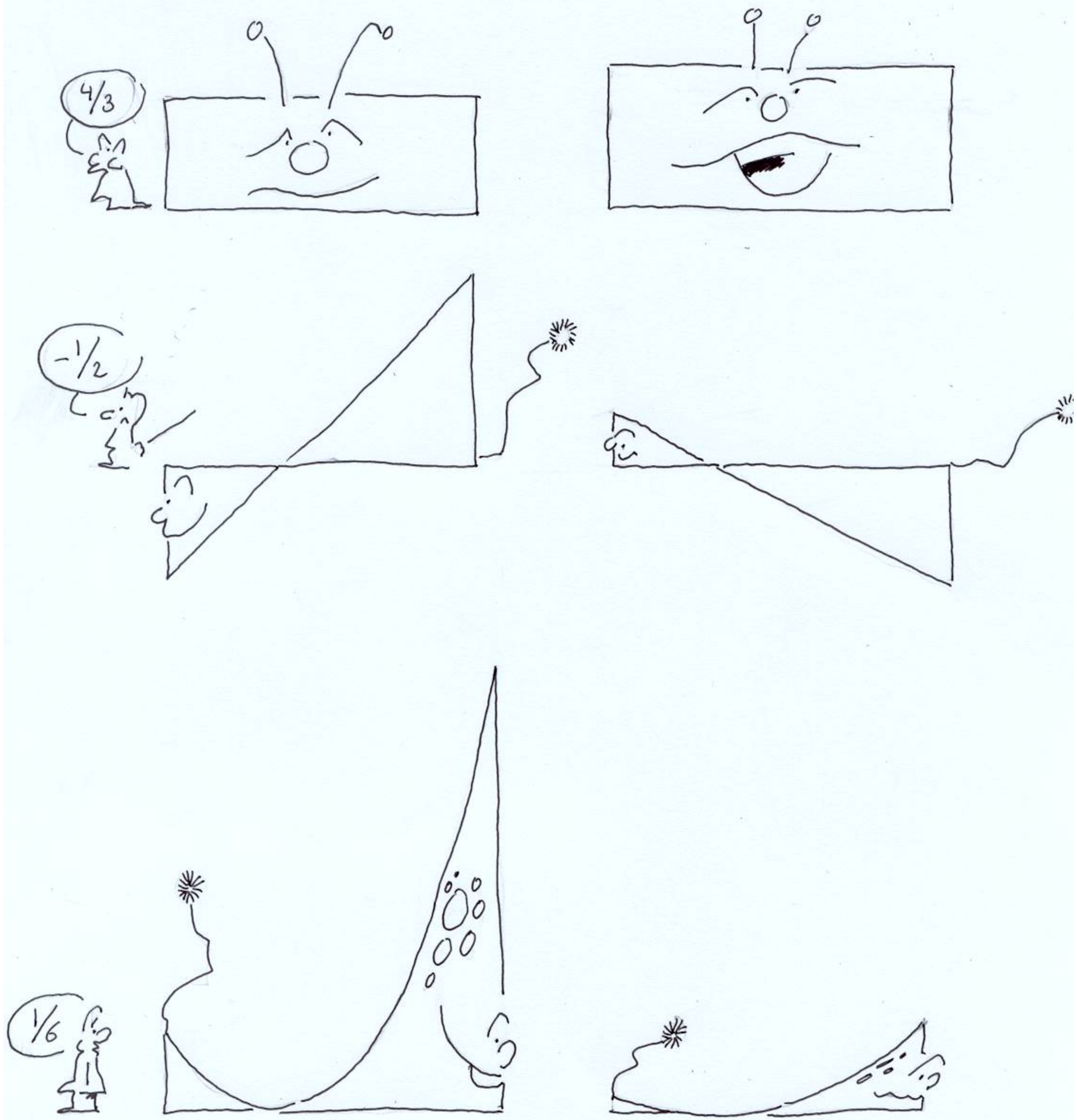
Allons chercher quelques monstres apprivoisés…



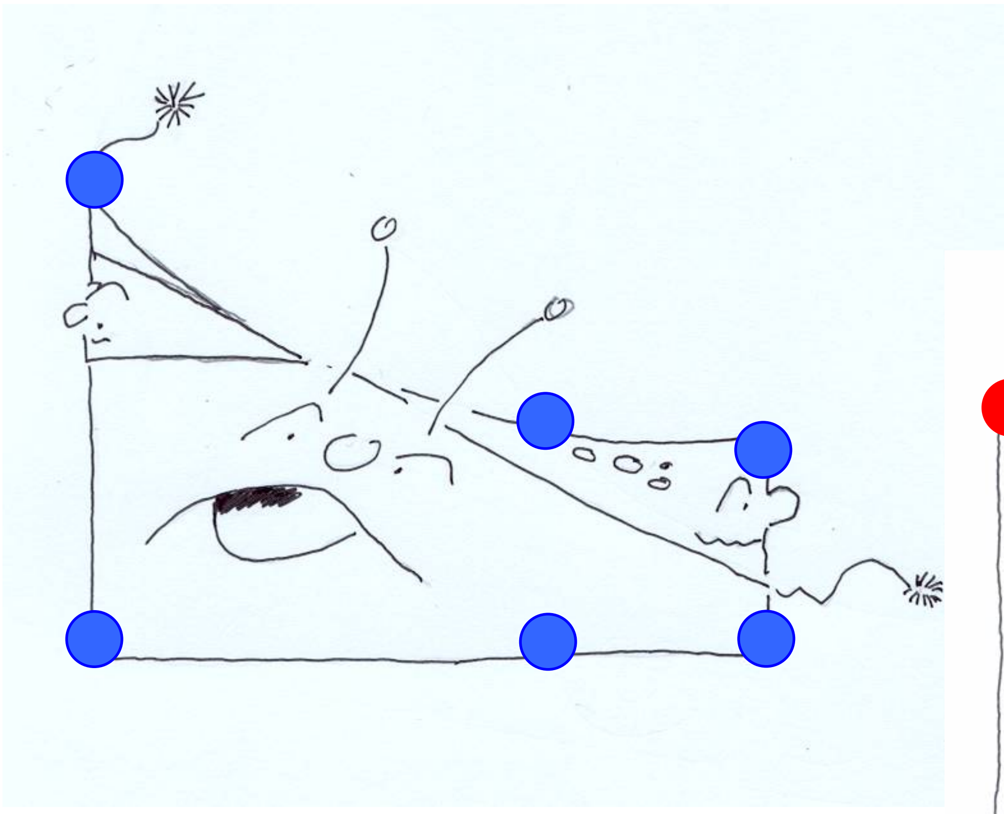
Les 3 fonctions de base d'espace discret de dimension 3.

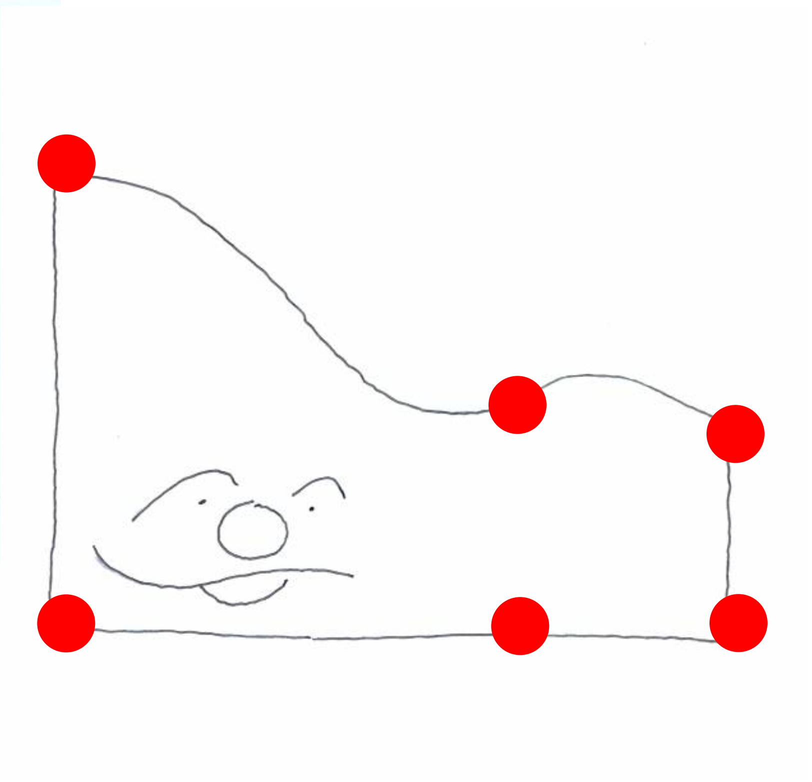
Base d'un espace vectoriel

Etes-vous un bon dresseur ?

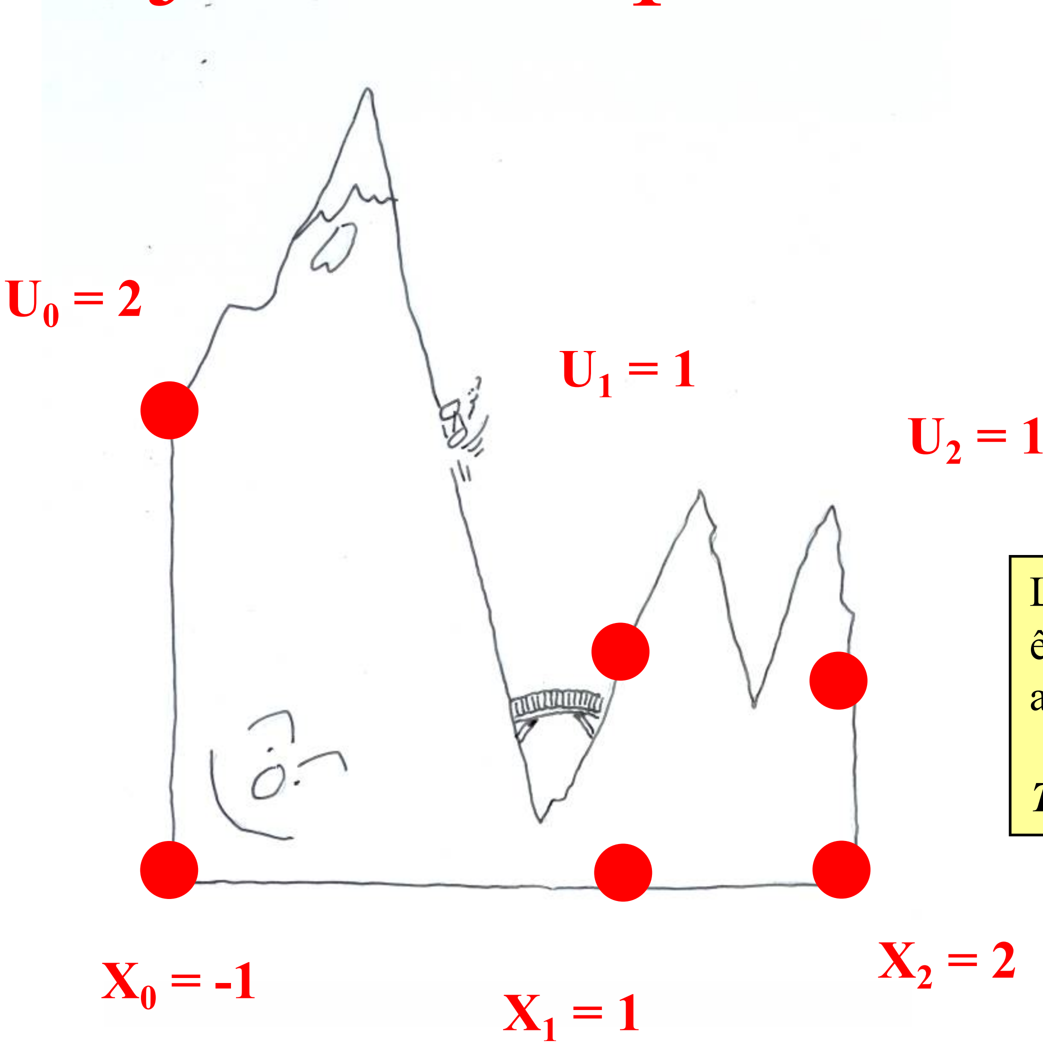


Et la phase critique…





Et si j'avais l'esprit montagnard…



La prise de mesures ne peut pas être réalisée de manière arbitraire…

Théorie de l'échantillonnage

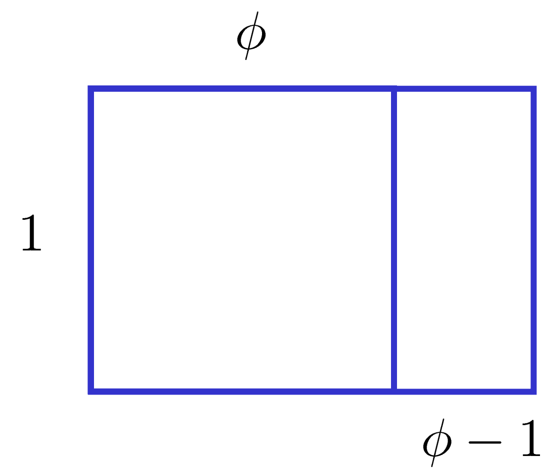
Le zéro problème

$$\begin{cases} x_1 + x_2 &= 1 \\ \phi x_1 + (1-\phi) x_2 &= 1 \end{cases}$$

Nombre d'or

The Golden Ratio

```
from math import sqrt

phi = (1.0 + sqrt(5.0)) / 2.0
print(phi)
```



$$\frac{1}{\phi} = \frac{\phi - 1}{1}$$

$$\downarrow$$

$$\phi^2 - \phi - 1 = 0$$

vl — python — 77×9

```
bash-3.2$ python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct  6 2017, 12:04:38)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from math import sqrt
>>> phi = (1.0 + sqrt(5.0)) / 2.0
>>> print(phi)
1.618033988749895
>>> 
```

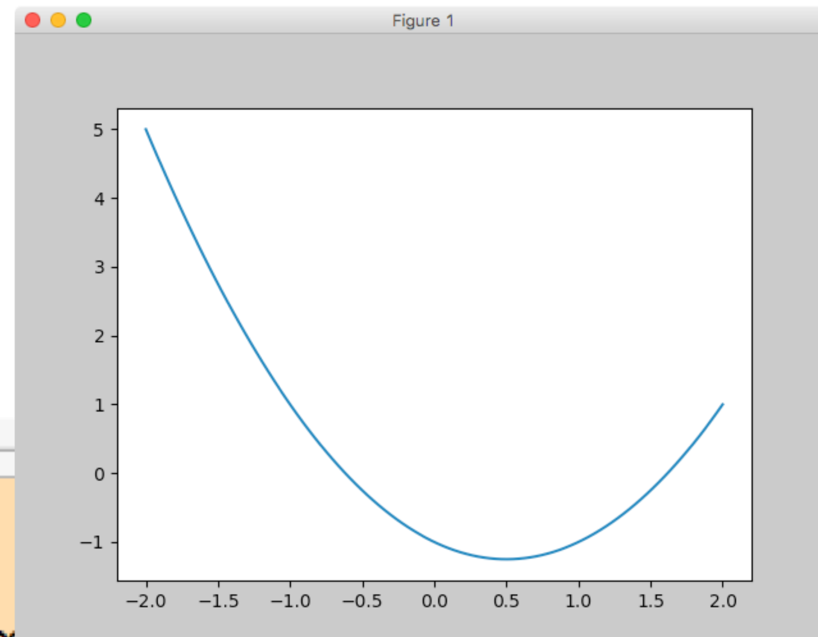
Polynômes avec Python

$$\phi^2 - \phi - 1 \doteq 0$$

```
x = linspace(-2,2,100)
plt.plot(x,polyval(p,x))
plt.show()
```

$$\phi = \frac{1 \pm \sqrt{5}}{2}$$

```
p = [1,-1,-1]
r = roots(p)
print(r)
```



vl — python goldenratio.py —

```
bash-3.2$ python goldenratio.py
 -0- Golden ratio number
1.618033988749895
 -1- Golden ratio number as roots of po
[ 1.61803399 -0.61803399]
```

Calcul symbolique : sympy

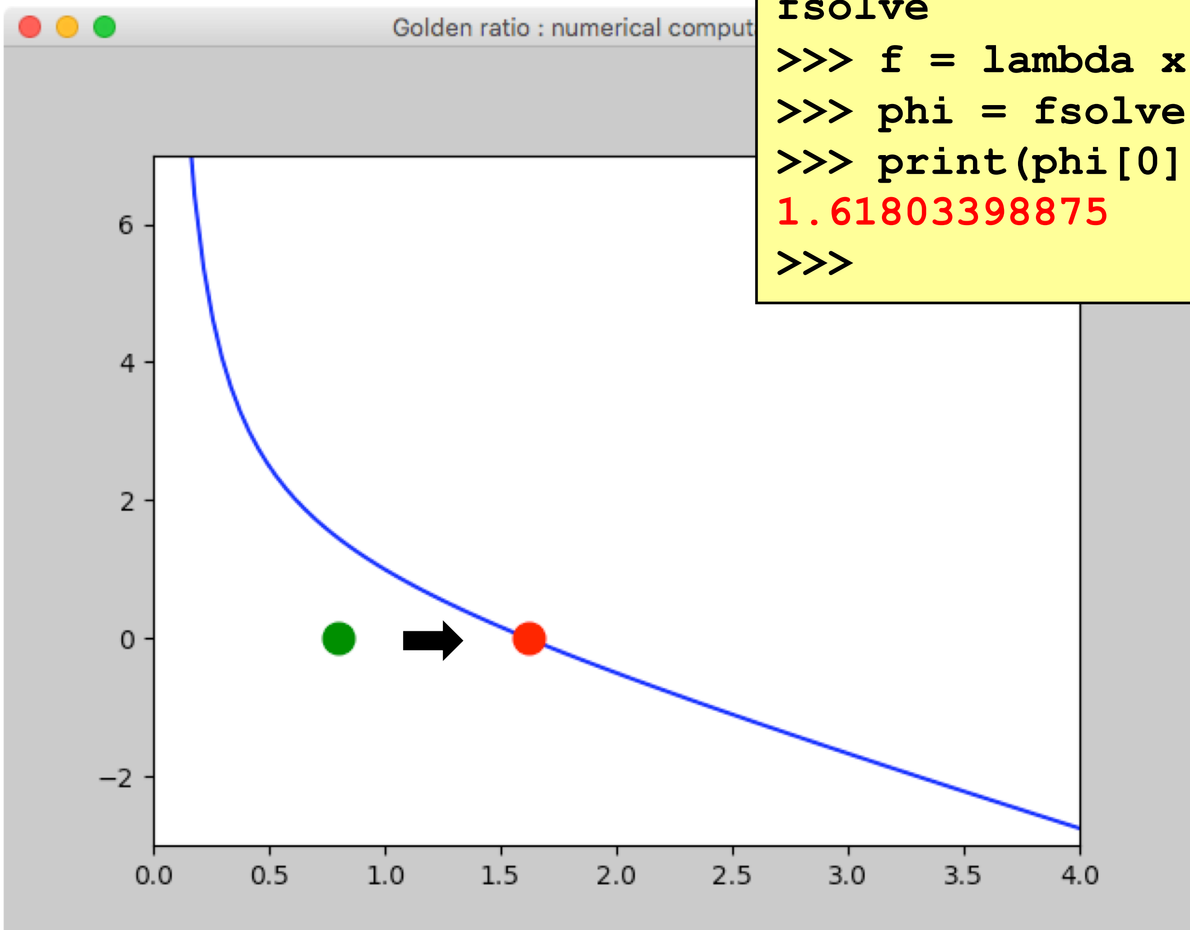
$$\phi^2 - \phi - 1 \doteq 0$$

```
>>> from sympy import symbols,solve,evalf
>>> x = symbols('x')
>>> r = solve(1/x-x+1,x)
>>> print(r)
[1/2 + sqrt(5)/2, -sqrt(5)/2 + 1/2]
>>> phi = r[0]
>>> print(phi)
1/2 + sqrt(5)/2
>>> print(phi.evalf(50))
1.6180339887498948482045868343656381177203091798058
>>> print(phi.evalf())
1.61803398874989
```

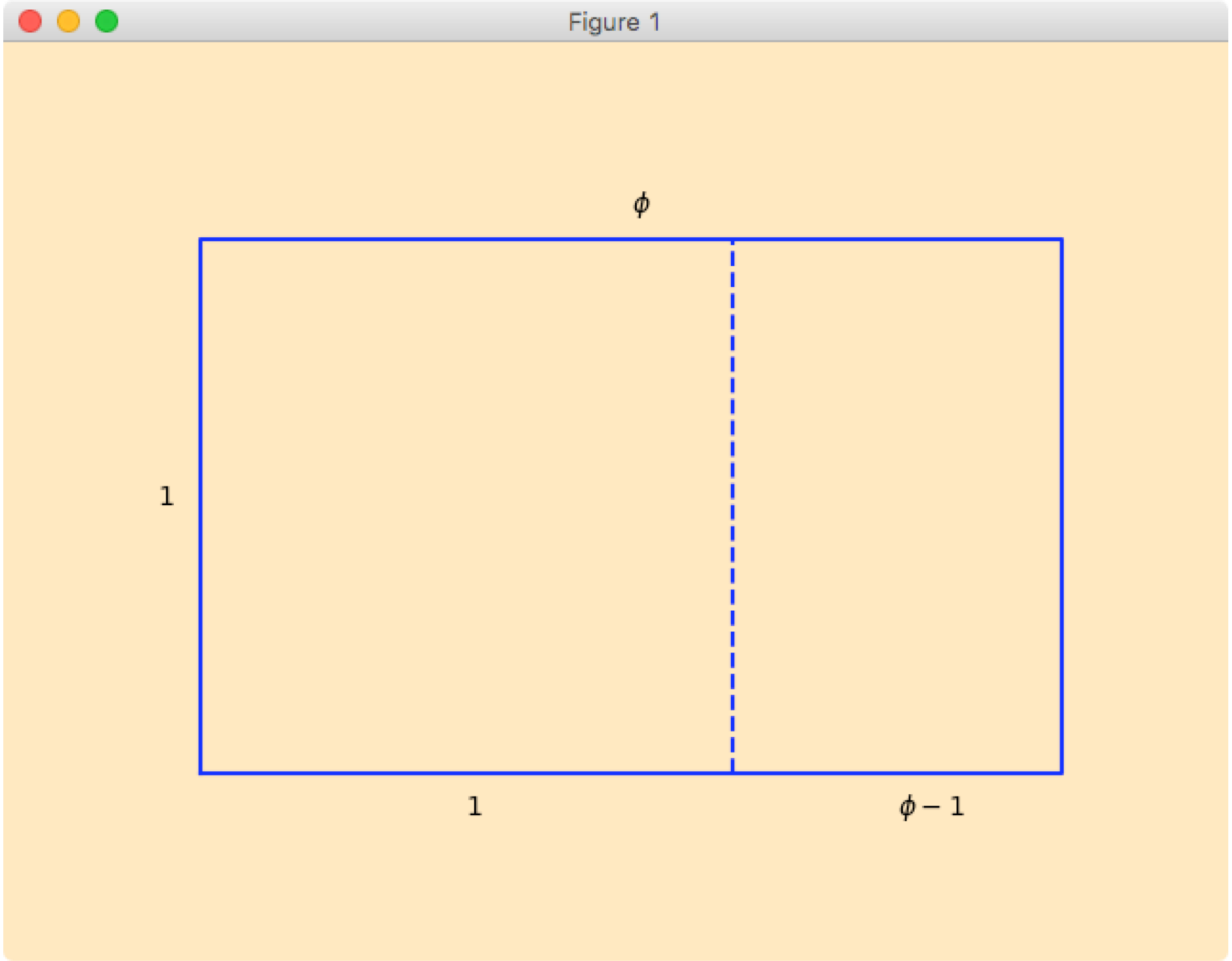
$$\phi = \frac{1 \pm \sqrt{5}}{2}$$

Résolution numérique : scipy

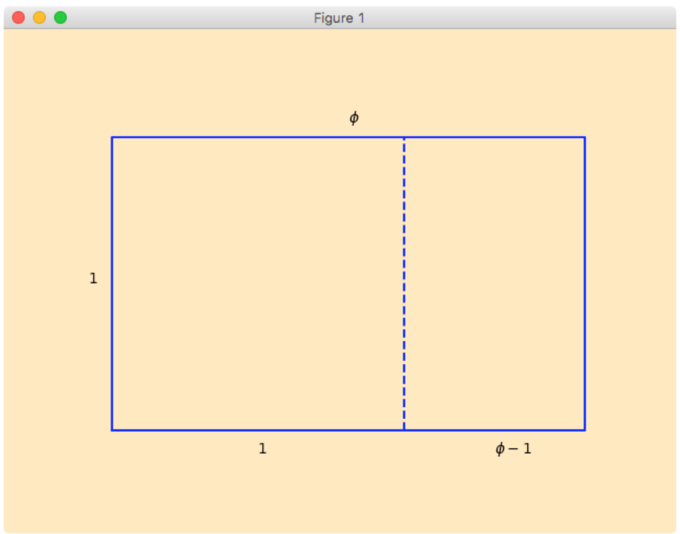
```
>>> from scipy.optimize import
fsolve
>>> f = lambda x : 1/x - (x-1)
>>> phi = fsolve(f,0.8)
>>> print(phi[0])
1.61803398875
>>>
```



Faire des jolis dessins…



py-files




```
def goldRect():

   """ plots the golden rectangle  """

   phi = (1 + sqrt(5.0)) / 2
   x = [0,phi,phi,0,0]
   y = [0,0,1,1,0]
   u = [1,1]
   v = [0,1]

   plt.axis('equal')
   plt.axis('off')
   plt.text(phi/2,1.1,'$\phi$')
   plt.text((1+phi)/2,-0.1,'$\phi - 1$')
   plt.text(-0.1, 0.5,'1')
   plt.text( 0.5,-0.1,'1')
   plt.plot(x,y,'-b')
   plt.plot(u,v,'--b')
   plt.show()
```

LaTeX commands

```
>>> from goldenRatioRectangle import goldRect
>>> goldRect()
>>> help(goldRect)
```

Fractions continues…

$$\frac{1}{\phi} = \frac{\phi - 1}{1}$$

$$1 + \frac{1}{\phi} = \phi$$

$$1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\cdots}}}} = \phi$$

```
>>> from math import sqrt
>>> n = 6
>>> p = '1'
>>> for k in range(n):
...     p = '1+1/(' + p + ')'
...
>>> print(p)
1+1/(1+1/(1+1/(1+1/(1+1/(1+1/(1))))))
>>> phi = eval(p)
>>> print(phi)
1.6153846153846154
>>> err = (1+sqrt(5))/2 - phi
>>> print(err)
0.0026493733652794837
```

La variable **`p`** est une chaîne de caractères générée en commençant par un seul `'1'` et en encadrant à plusieurs reprises cette chaîne avec `'1+1/('` à l'avant et `')'` à l'arrière. Quelle que soit la longueur de cette chaîne, il s'agit d'une expression Python valide.

Une autre implémentation…

$$\frac{1}{\phi} = \frac{\phi - 1}{1}$$

$$\downarrow$$

$$1 + \frac{1}{\phi} = \phi$$

$$\downarrow$$

$$\frac{\phi_n + 1}{\phi_n} = \phi_{n+1}$$

```
>>> from math import sqrt
>>> n = 6
>>> p = 1
>>> q = 1
>>> for k in range(n):
...     s = p
...     p = p + q
...     q = s
...
>>> phi = eval('%d/%d' % (p,q))
>>> print(phi)
1.6153846153846154
>>> err = (1 + sqrt(5))/2 - phi
>>> print(err)
0.0026493733652794837
```