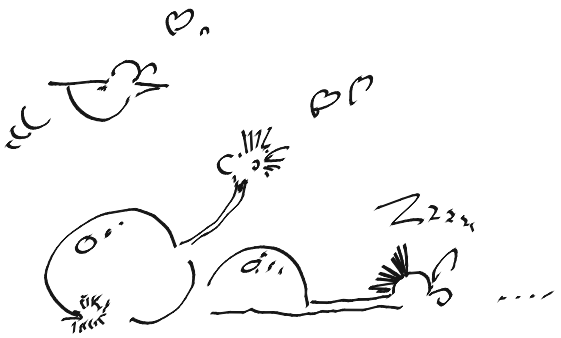
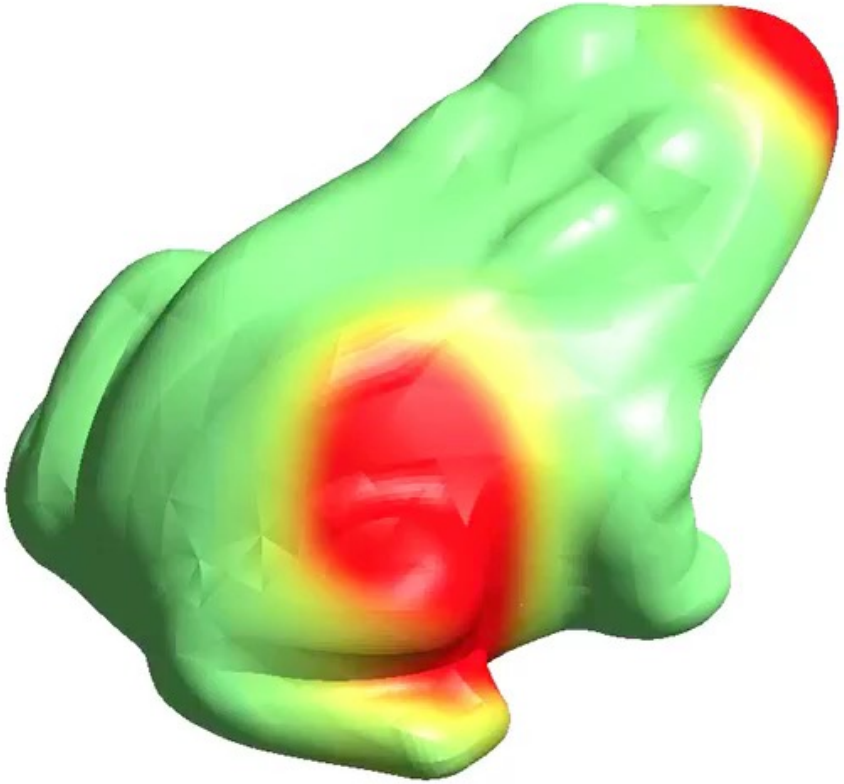
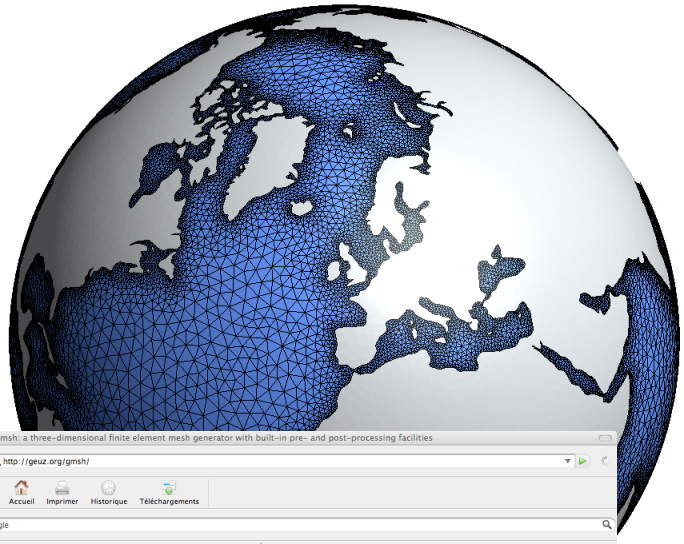
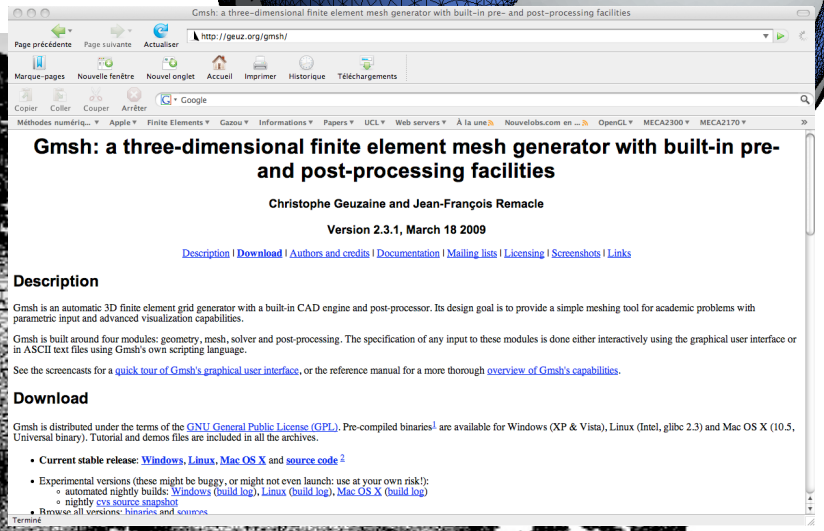
Les méthodes numériques
A quoi cela sert-il ?





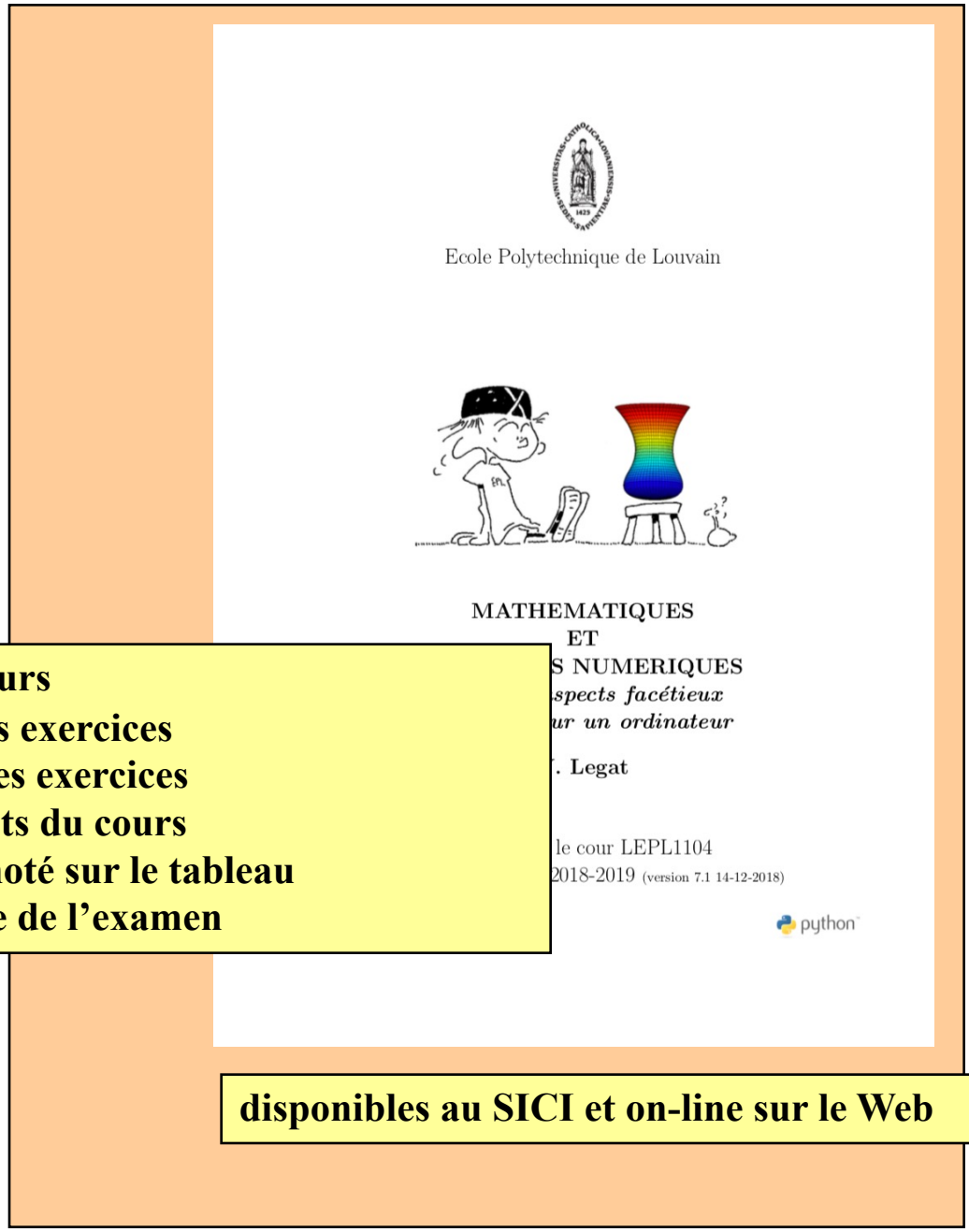
Second-generation Louvain-la-Neuve Ice-ocean Model





1.8 millions de triangles,

Notes de cours



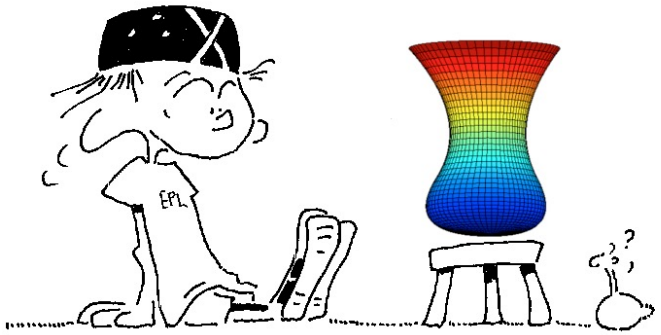
Notes de cours
Enoncés des exercices
Solutions des exercices
Transparents du cours
Ce qui est noté sur le tableau
= la matière de l'examen

disponibles au SICI et on-line sur le Web

Livres de références…

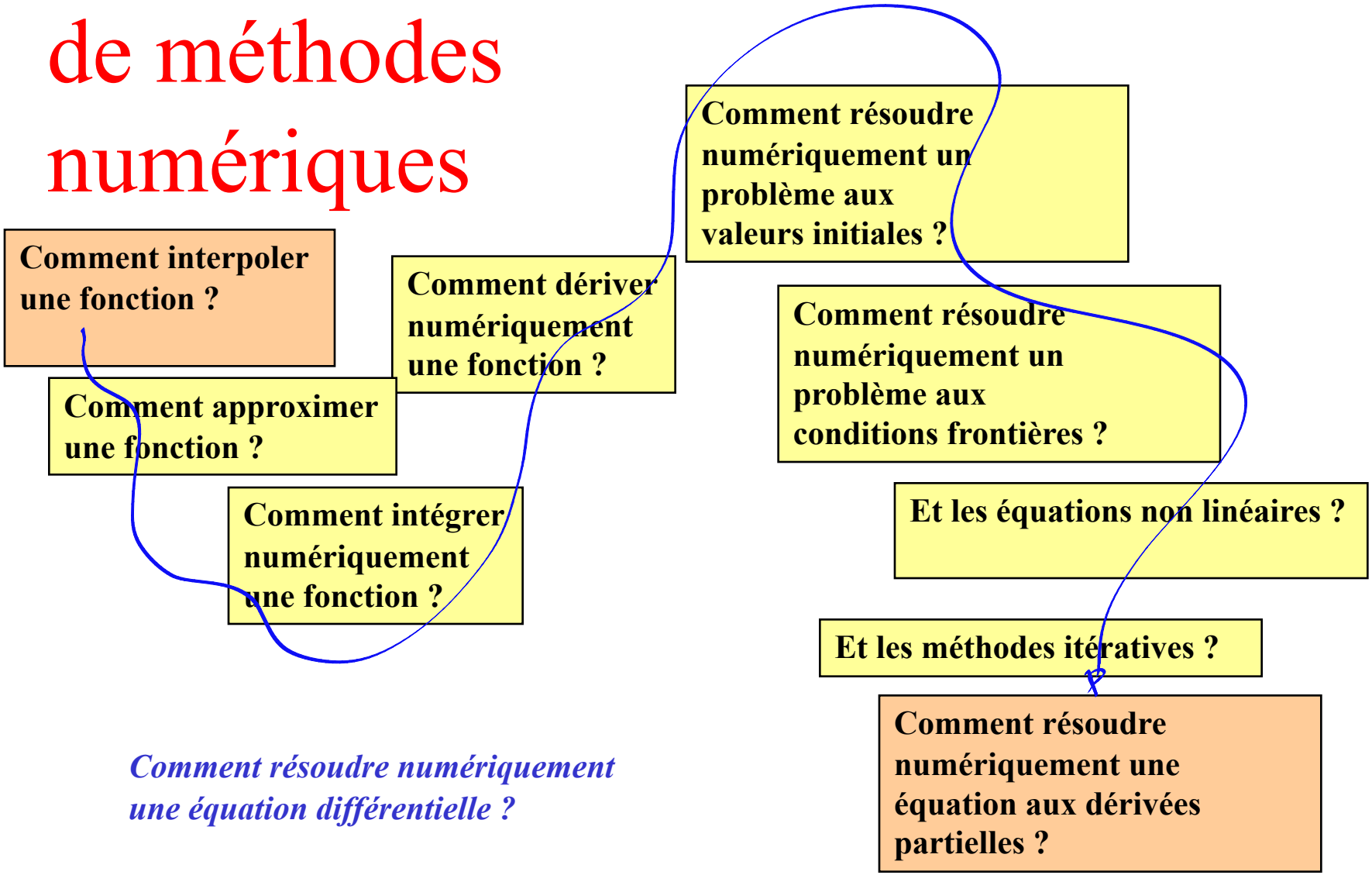
- Charles F. Van Loan, *Introduction to Scientific Computing*, Second Edition, Prentice Hall, Upper Saddle River, ISBN 0-13949157-0 (1999).
- Jacques Rappaz, Marco Picasso, *Introduction à l'analyse numérique*, Presses polytechniques et universitaires romandes, Lausanne, ISBN 2-88074363-X (2000).
- André Fortin, *Analyse numérique pour ingénieurs*, Seconde Edition, Presses internationales polytechniques, Montréal, ISBN 2-55300936-4 (2001).
- William L. Briggs, Van Emden Henson, Steve F. McCormick, *A Multigrid Tutorial*, Second Edition, SIAM, Philadelphia, ISBN 0-89871462-1 (2000).
- Brigitte Lucquin, Olivier Pironneau, *Introduction to Scientific Computing*, John Wiley & Sons, New York, ISBN 0-47197266-X (1998).
- Alfio Quarteroni, Fausto Saleri, *Scientific Computing with MATLAB*, Springer-Verlag, Berlin, ISBN 3-35044363-0 (2003).
- Desmond J. Higham, Nicholas J. Higham *Matlab Guide*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, ISBN 0-89871469-9 (2000).
- Michael T. Heath *Scientific Computing : an Introduction Survey*, McGraw Hill, New-York, ISBN 0-07-115336-5 (1997).
- K. E. Atkinson, *An Introduction to Numerical Analysis*, Second Edition, John Wiley & Sons, New York (1989).
- S. D. Conte, C. de Boor, *Elementary Numerical Analysis, An Algorithmic Approach*, Third Edition, McGraw-Hill Book Company, New York (1980).
- B.M. Irons, N.G. Shrive, *Numerical Methods in Engineering and Applied Sciences : numbers are fun*, Second Edition, John Wiley and Sons (1987).
- John H. Mathews, *Numerical Methods for Mathematics, Science and Engineering*, Second Edition, Prentice Hall, Englewood Cliffs, ISBN 0-13624990-6 (1992).
- W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition, Cambridge University Press, Cambridge (1994).

Comment contacter le titulaire du cours?



- **Il n'est pas possible de prendre rendez-vous auprès de moi.**
- **Il n'est pas possible de me consulter par courriel.**
- **Si vous me trouvez dans mon bureau et que mon humeur volage est positive, je vous consacrerai un peu de temps (et même parfois beaucoup…)**
- **Si il est indiqué sur ma porte ne pas déranger, mon humeur volage est négative. Il ne faut donc pas me déranger, car votre demande risque de ne pas être traitée avec toute la douceur requise.**

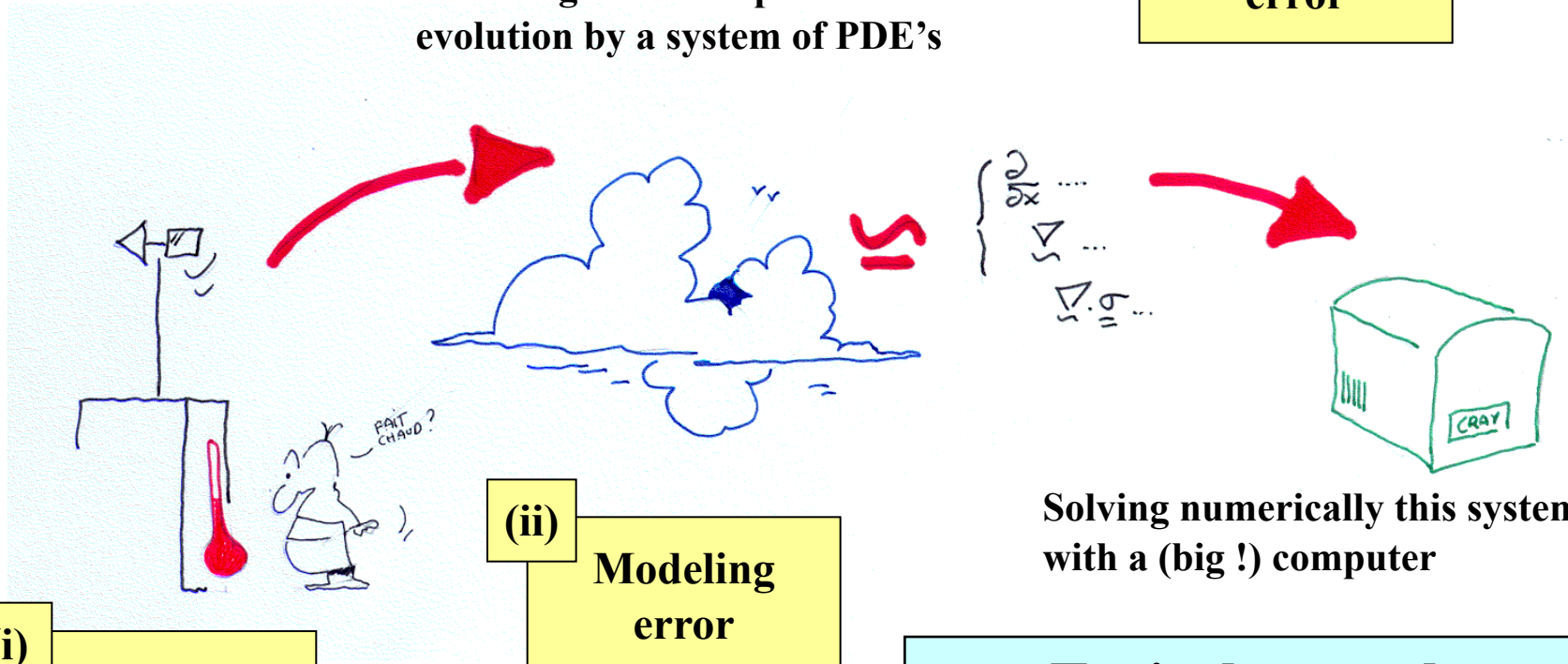
Plan des cours de méthodes numériques



Weather forecasts are not always reliable…

(iii) Computation error

Modeling the atmosphere evolution by a system of PDE's



(ii) Modeling error

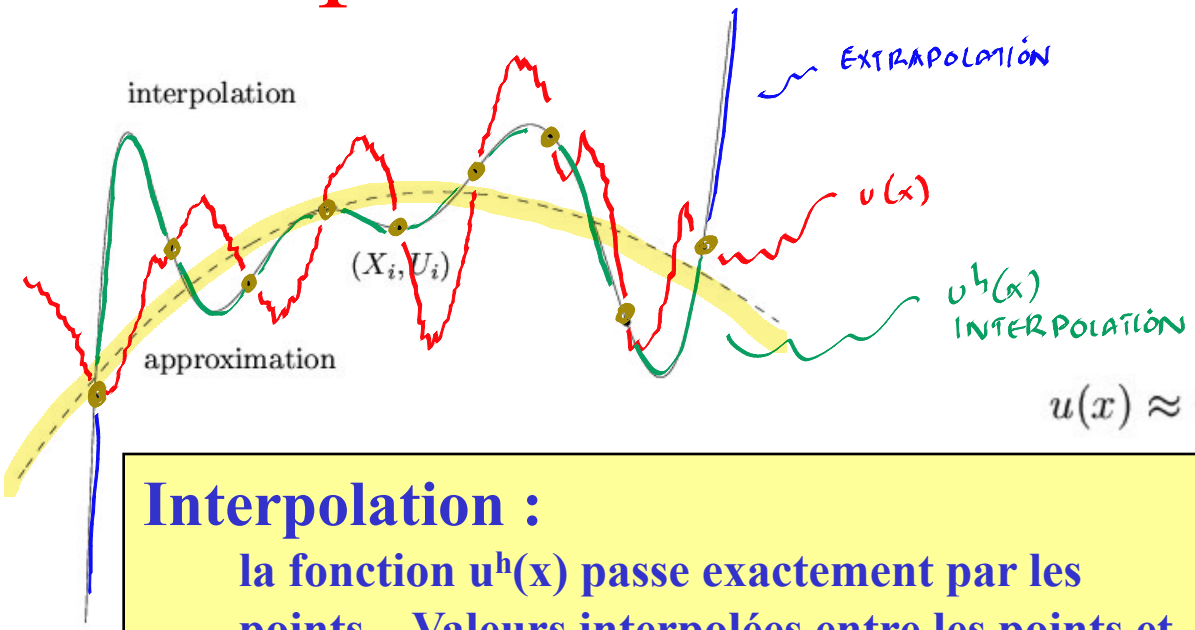
Solving numerically this system with a (big !) computer

(i) Data error

Measuring the current weather

Typical example of mathematical modeling and computer simulation

Interpolation, approximation et extrapolation…



Fonctions de base spécifiées a priori

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

Paramètres inconnus

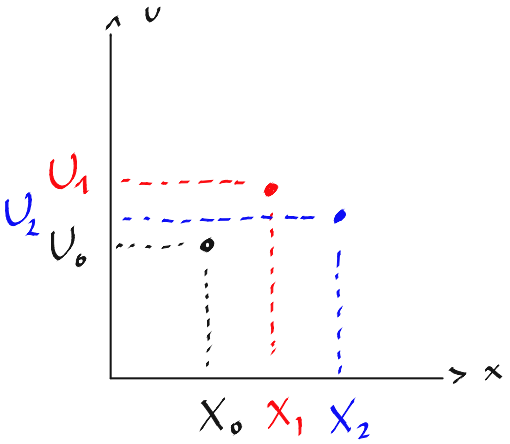
Interpolation :

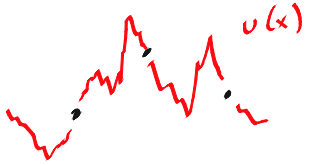
la fonction $u^h(x)$ passe exactement par les points. Valeurs interpolées entre les points et valeurs extrapolées hors de l'intervalle.

Approximation :

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

Matrice de Vandermonde





$v \in U$

$\dim U = \infty$

$v^h(x) = ax^2 + bx + c$

$v^h \in U^h$

$\dim(U^h) = 3$

$$v(x) \simeq v^h(x) = a_0 \varphi_0(x) + a_1 \varphi_1(x) + a_2 \varphi_2(x)$$

$a_0 \in \mathbb{R}$ INCONNUS

$\varphi_0 \in U^h$ FIXES / CONNUS A PRIORI

? a_0, a_1, a_2

tel que

$v(X_0) = v^h(X_0)$

U_0

$U_1 = v^h(X_1)$

$U_2 = v^h(X_2)$

EXEMPLE

$\varphi_0(x) = 1$

$\varphi_1(x) = x$

$\varphi_2(x) = x^2$

? a_0, a_1, a_2

tel que

$$\underbrace{u(x_0)}_{U_0} = u^h(x_0)$$

$U_1 = u^h(x_1)$

$U_2 = u^h(x_2)$

$$\begin{cases} a_0\varphi_0(x_0) + a_1\varphi_1(x_0) + a_2\varphi_2(x_0) = U_0 \\ a_0\varphi_0(x_1) + a_1\varphi_1(x_1) + a_2\varphi_2(x_1) = U_1 \\ a_0\varphi_0(x_2) + a_1\varphi_1(x_2) + a_2\varphi_2(x_2) = U_2 \end{cases}$$

$$\underbrace{\begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) \\ & \cdots & \\ \varphi_0(x_2) & & \varphi_2(x_2) \end{bmatrix}}_{\underline{\underline{A}}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}}_{\underline{x}} = \underbrace{\begin{bmatrix} U_0 \\ U_1 \\ U_2 \end{bmatrix}}_{\underline{U}}$$

VECTEURS MATRICE EN GRAS

EXEMPLE

$\varphi_0(x) = 1$

$\varphi_1(x) = x$

$\varphi_2(x) = x^2$

? a_i

$$\sum_{i=0}^{n} a_i \varphi_i(x_j) = U_j$$

$j = 0 \dots n$

PROBLEME DE L'INTERPOLATION

$\underline{\underline{A}} \cdot \underline{x} = \underline{U}$

FORGET ALGEBRA!

$\underline{x} = \underline{\underline{A}}^{-1} \cdot \underline{U}$ NO!

Et ce qu'on trouve dans le syllabus

Problème de l'interpolation

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

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

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

Cas particulier : Interpolation polynomiale

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

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

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

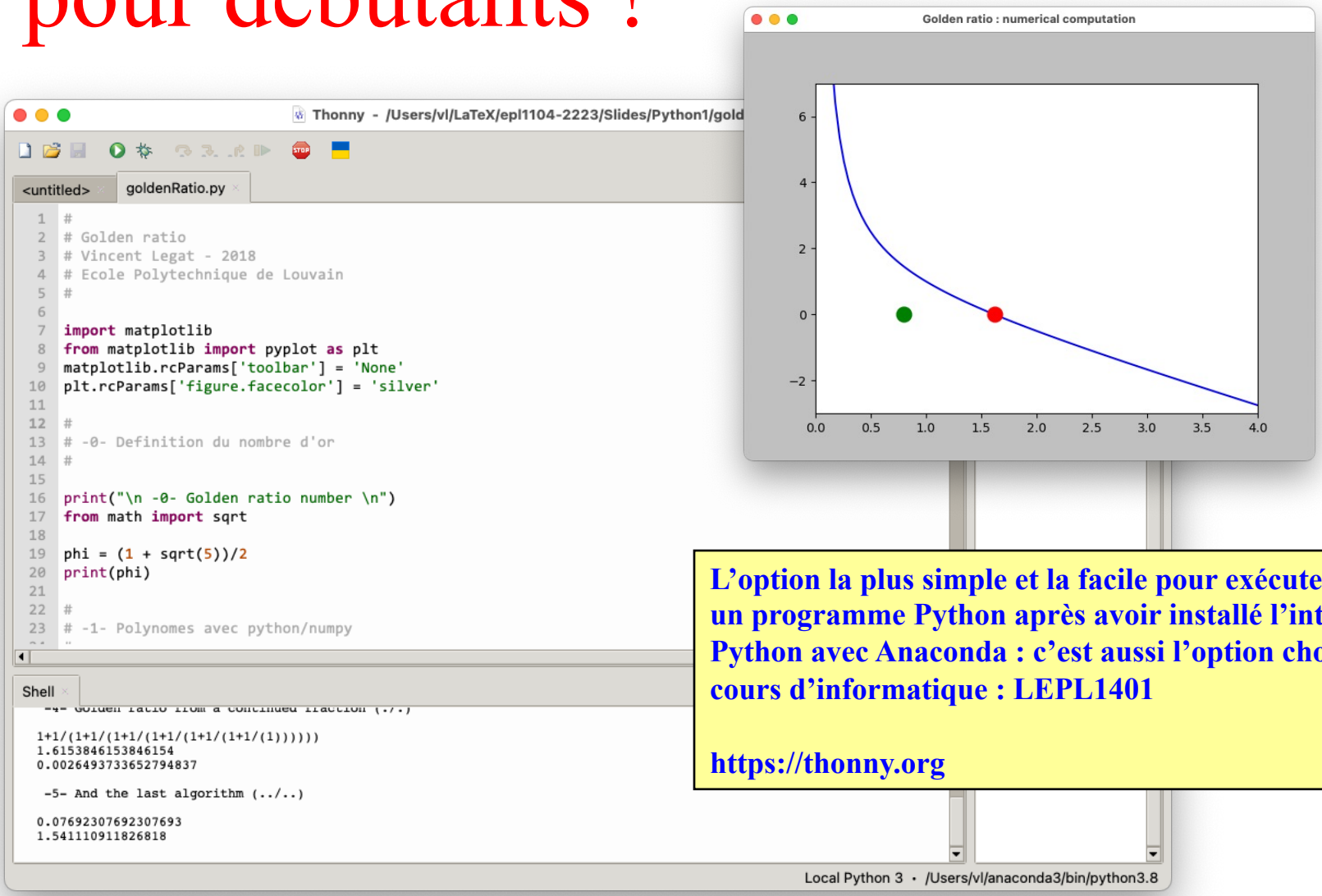
Né le 28 février 1735 à Paris
Décédé le 1er janvier 1796 à Paris

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

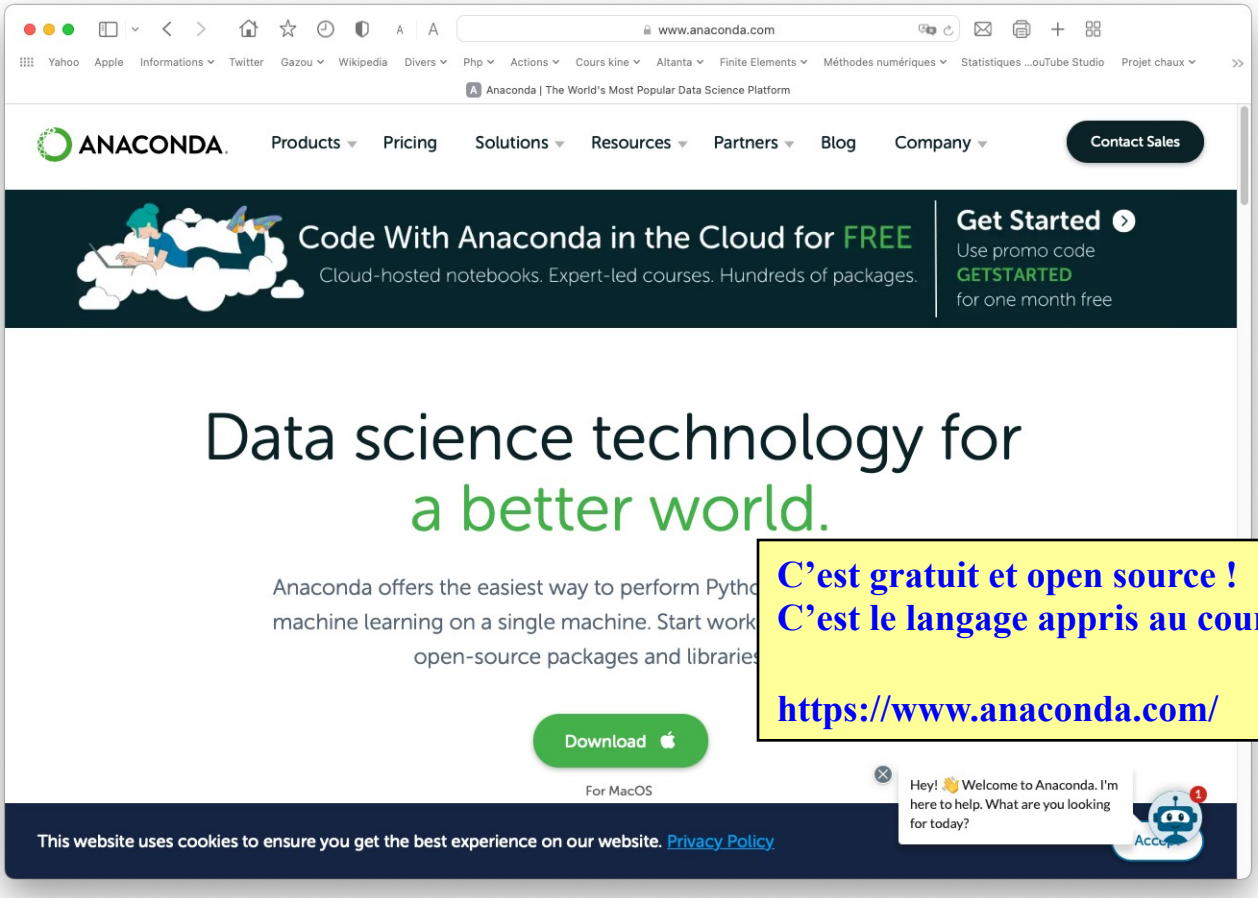
Un éditeur Python pour débutants !



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 ?

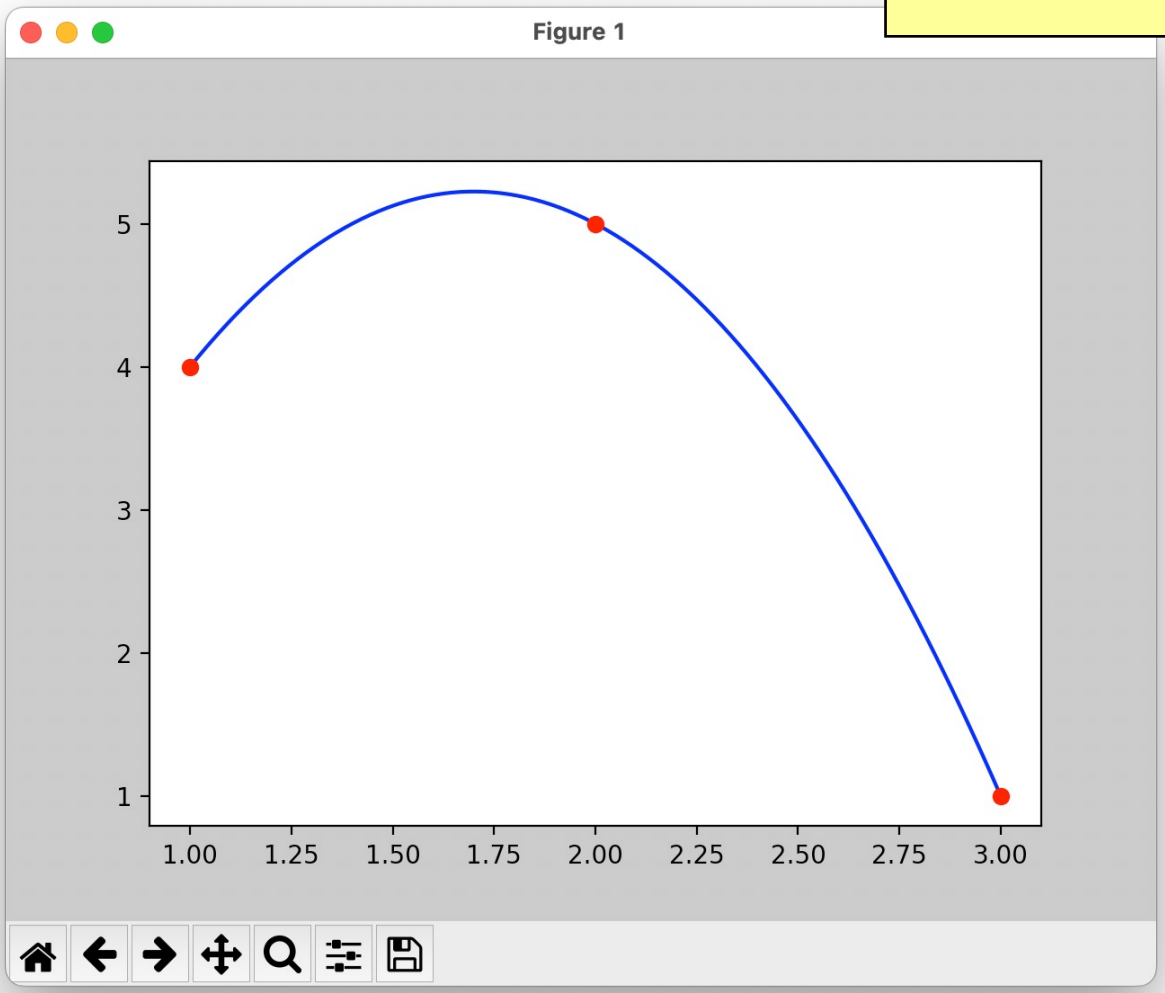


C'est gratuit et open source !
C'est le langage appris au cours d'informatique !

https://www.anaconda.com/

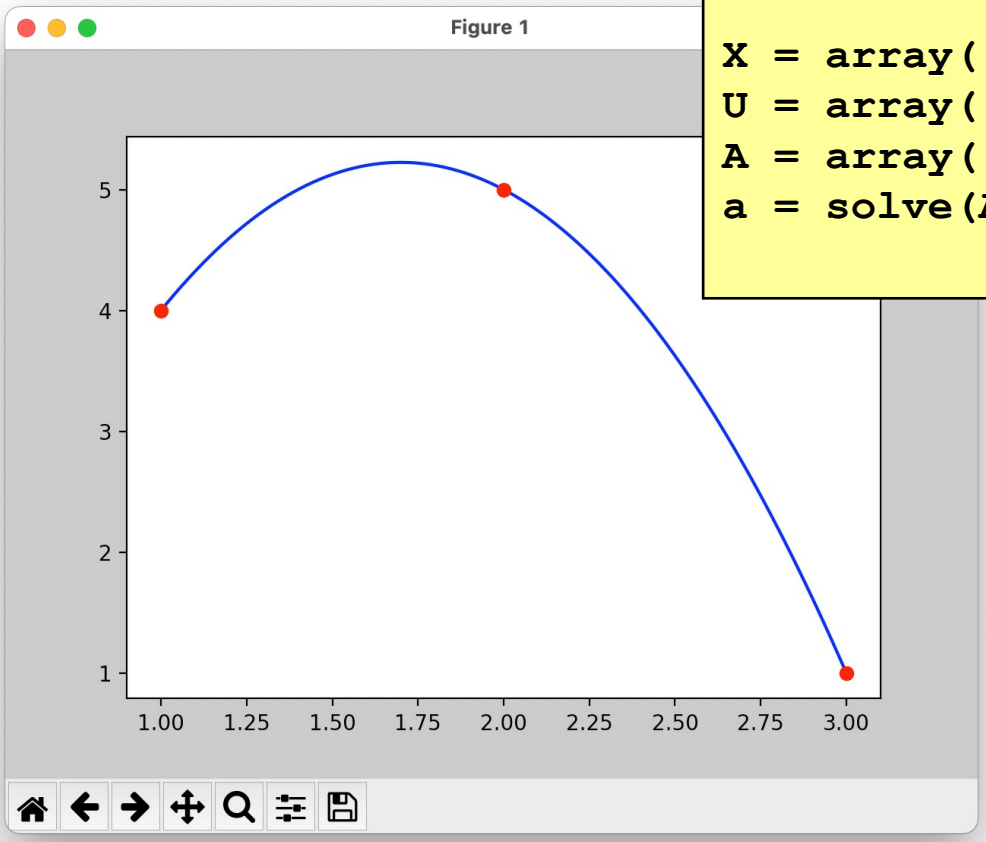
Faisons le calcul

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

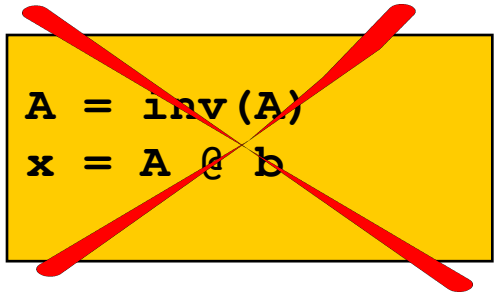


Faisons le calcul avec numpy

```
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...




```
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
```

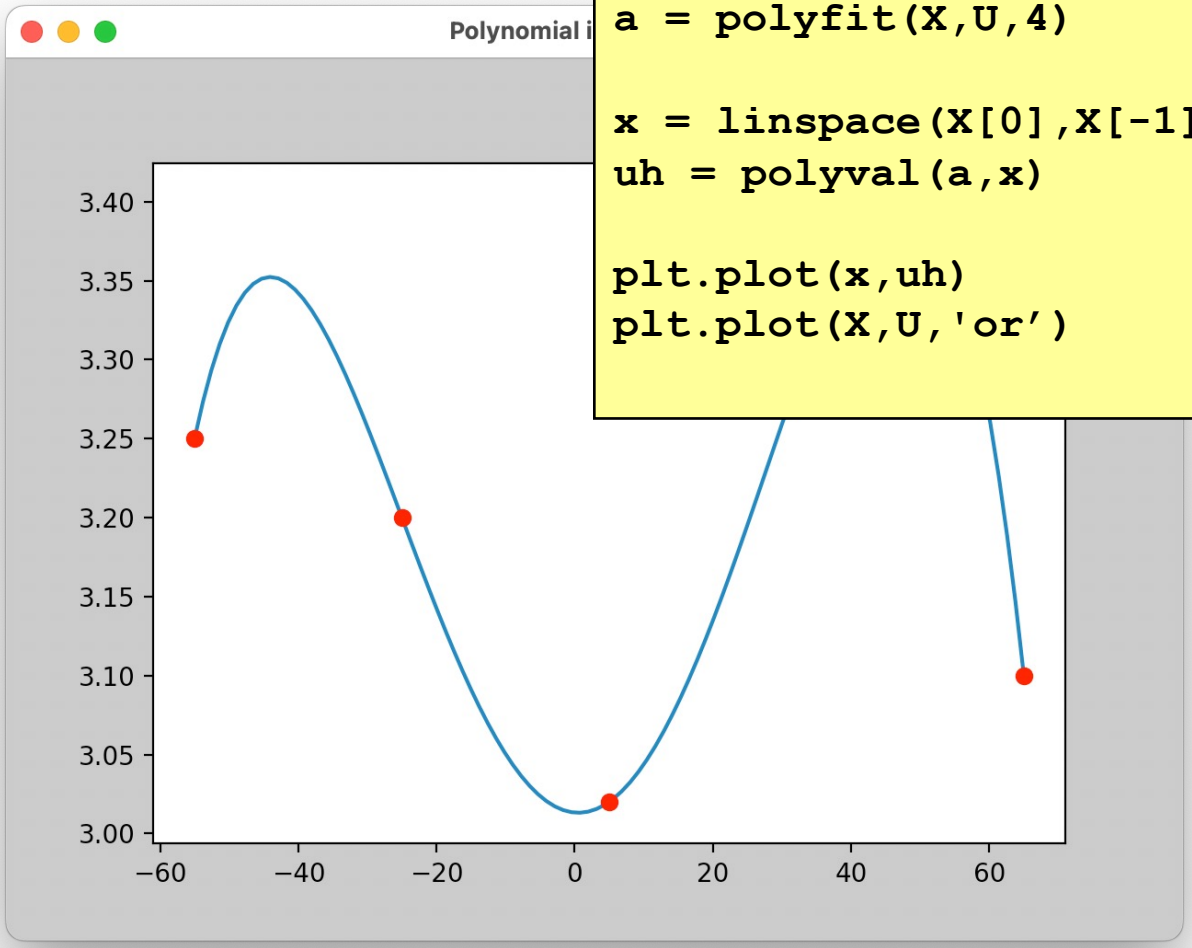
A better way, from both an execution time and numerical accuracy standpoint, is to use the matrix division operator

```
x = A\b
```

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

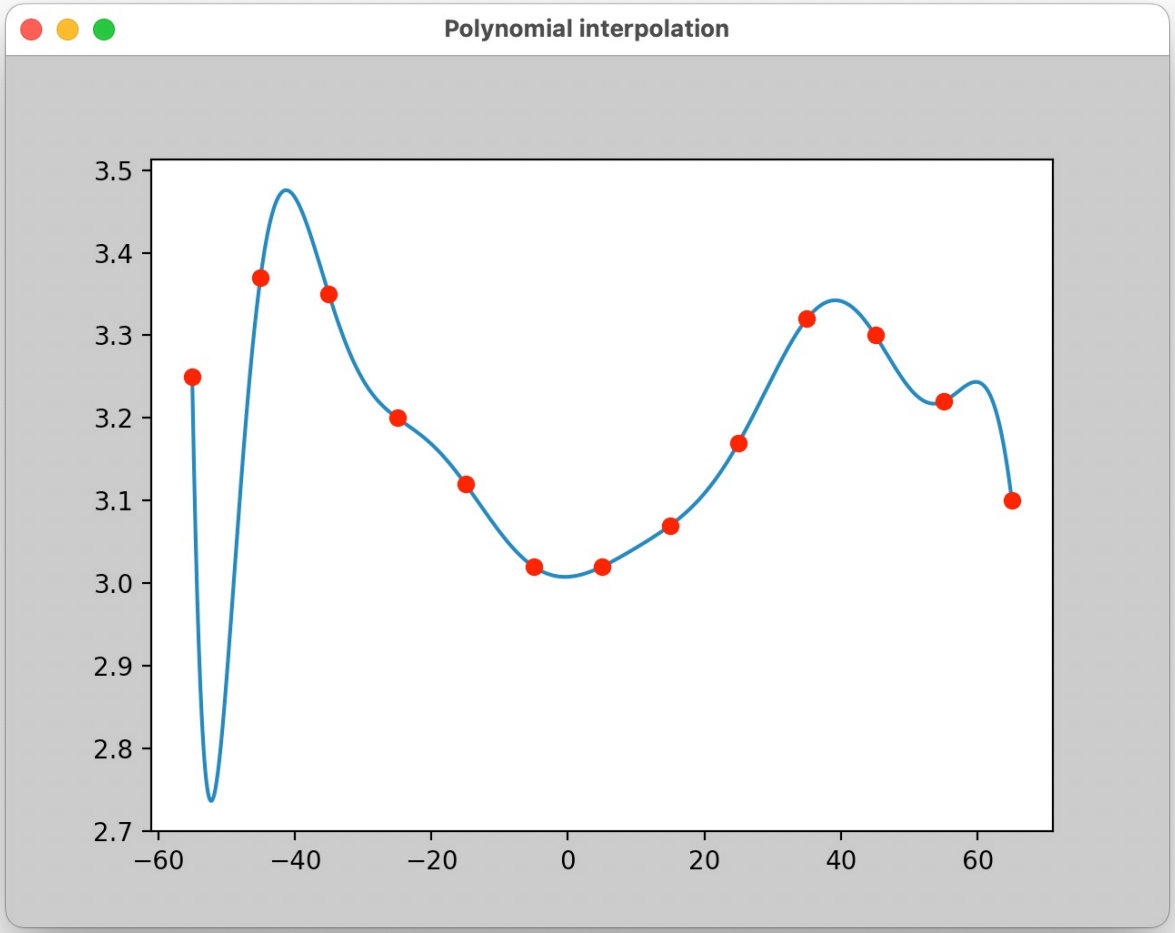
Exemple

```
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')
```

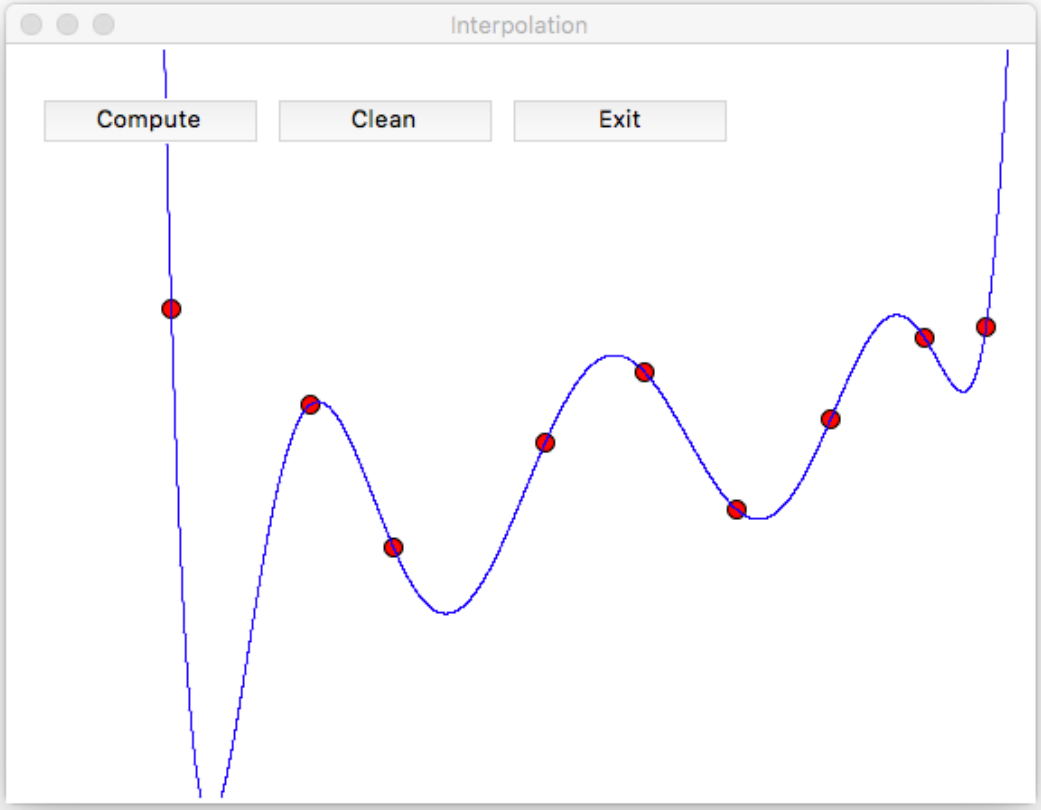


Davantage de données…

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

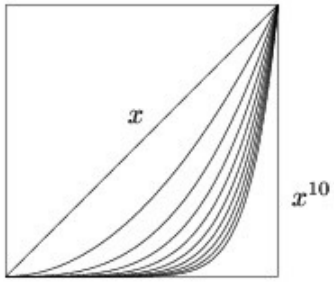


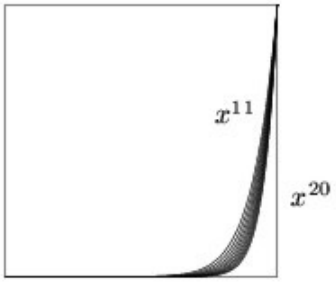
Le polynôme d'interpolation est un être bien turbulent…

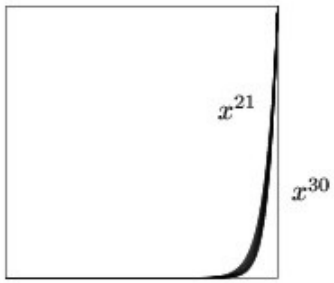


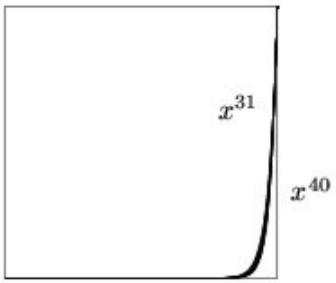
Les monômes

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









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

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

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

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

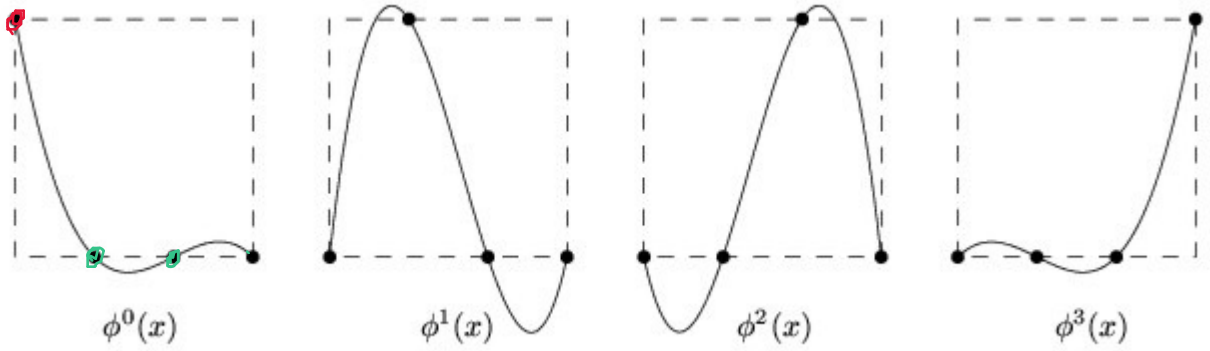
Solution perturbée : a=1.99999 et b=0.00001

Système bien conditionné

Comment savoir si un système est bien conditionné ?

```
vl — bash — 59×19
======= 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]))
```



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

Les fonctions de base de Lagrange

Idée…

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

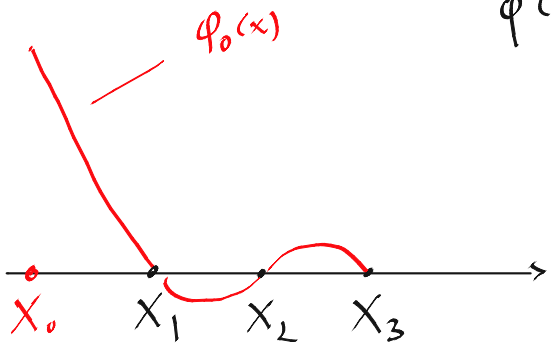
Choisir les fonctions de base afin que la matrice du système soit la matrice unité !

Même solution par unicité de l'interpolation !

Pas de problème de conditionnement !

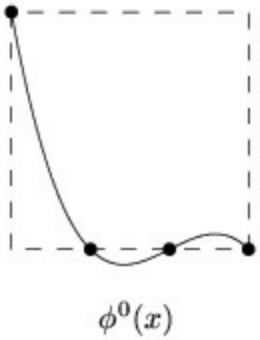
Polynomes de Lagrange

$$\varphi_0(x_0) = 1$$
$$\varphi_0(x_1) = 0$$
$$\varphi(x_2) = 0$$
$$\varphi(x_3) = 0$$



$$\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}$$

$\varphi_0(x)$



Les fonctions de base de Lagrange

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

Erreur d'interpolation

Fonction inconnue

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

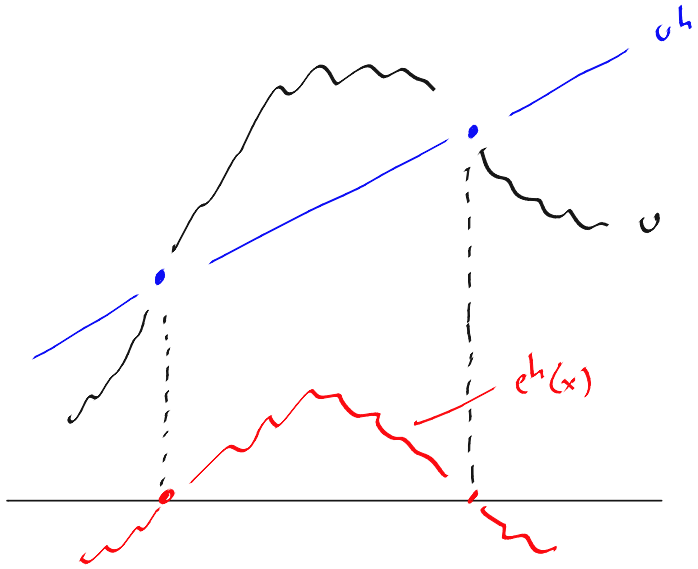
Interpolation polynomiale

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 :

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

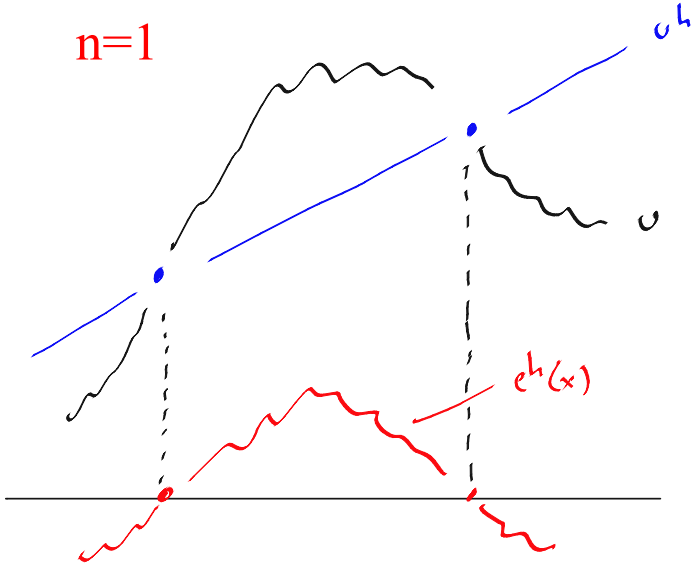
Erreur d'interpolation



$$\underbrace{u(x) - u_h(x)}_{e_h(x)} = \frac{u''(\xi(x))}{2!}(x - X_0)(x - X_1)$$

Démonstration

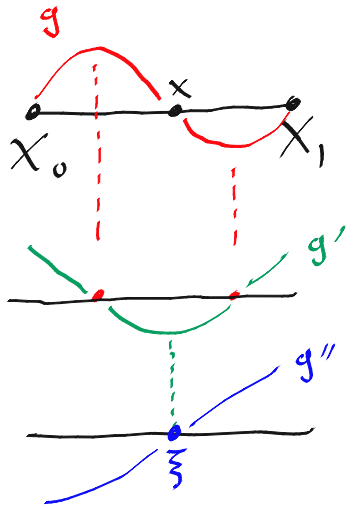
n=1



$x \in]x_0, x_1[$

$$g(t) = \underbrace{u(t) - u_h(t)}_{e^h(t)} - e^h(x)\,\frac{(t-x_0)(t-x_1)}{(x-x_0)(x-x_1)}$$

$t = x_0$	$= 0$	$= 0$
$t = x_1$	$= 0$	$= 0$
$t = x$	$= e^h(x)$	$= -e^h(x)$



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

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

$$u''(\xi) - \frac{e^h(x)\,2}{(x-x_0)(x-x_1)} = 0$$

□

$$e^h(x) = \frac{v''(\xi(x))}{2!}(x-x_0)(x-x_1)$$

$$|e^h(x)| \leq \left|\frac{v''(\xi(x))}{2!}\right| \left|(x-x_0)(x-x_1)\right|$$

$\left|\frac{v''(\xi(x))}{2!}\right| < \frac{C_2}{2}$

$f(x) \leq \frac{h^2}{4}$

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

$\mathcal{O}(h^2)$

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

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

$f'(x_m) = 0$

$\hookrightarrow x_m = h/2$

$f(x_m) = h^2/4$

VU COMME UNE FONCTION DE h POUR UN x FIXE

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

AU VOISINAGE DE ZERO

"$e^h(x)$" COMME UNE FONCTION DE h POUR UN PARAMETRE x

"$e_x(h)$" $f(h)$

And it is done !

The Big O notation :-)

ORDRE 2

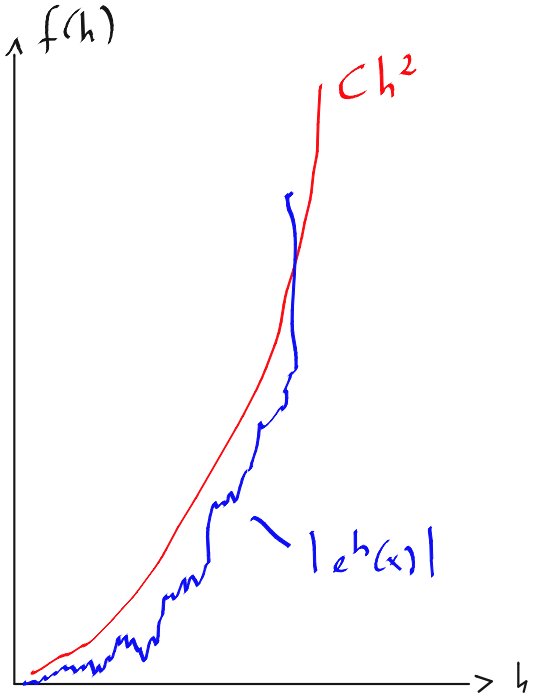
$$|e^h(x)| \leq \frac{C_2}{2}\frac{h^2}{4}$$

$$\mathcal{O}(h^2)$$

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 ZERO



$$\boxed{C\left[\frac{h}{2}\right]^m} = \frac{1}{2^m}\boxed{Ch^m}$$

ERREUR POUR $h/2$

ERREUR POUR h

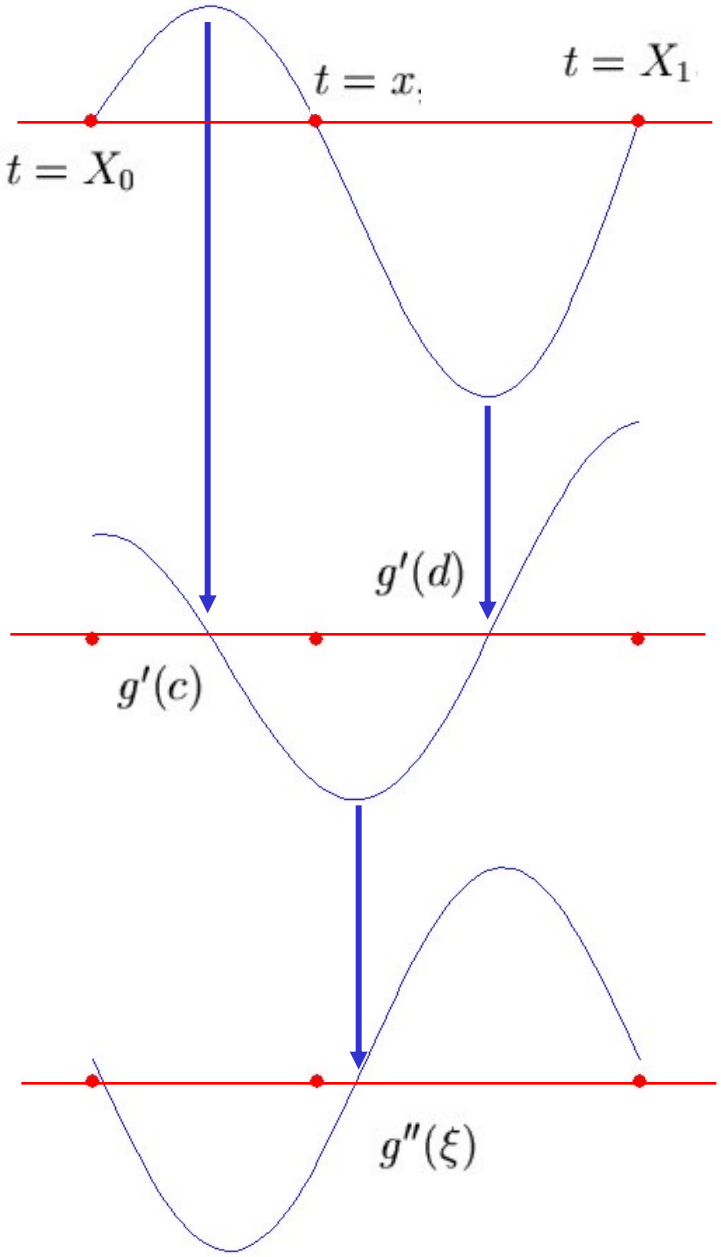
ORDRE		
	2	4
	3	8
	4	16
	5	32

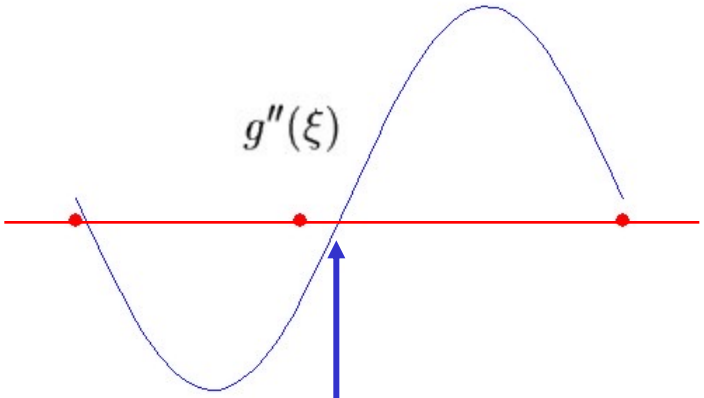
Nous allons démontrer uniquement le cas n=1
Ceci est le début de la démonstration :-)

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





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

$$\downarrow$$

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

$$\downarrow$$

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

Ceci est la fin de la démontration :-)

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

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

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

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

Borne d'erreur

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

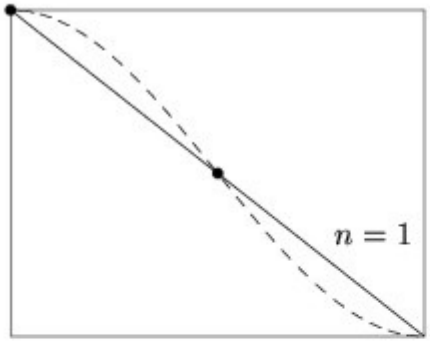
Définition 1.2. *Une approximation numérique dont le terme d'erreur est $\mathcal{O}(h^m)$ est dite d'ordre m.*

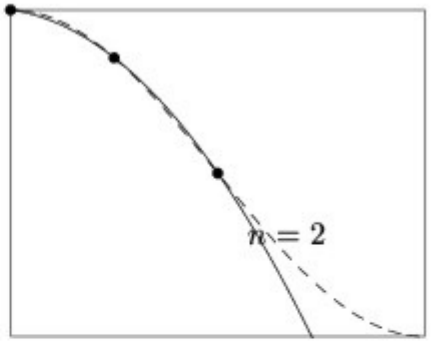
Polynôme de Taylor de degré n est souvent d'ordre n+1
Interpolation polynomiale par n+1 points équidistants est souvent d'ordre n+1

Signification intuitive de l'ordre d'une méthode numérique

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

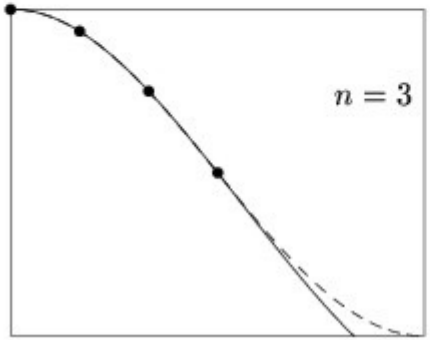
Convergence

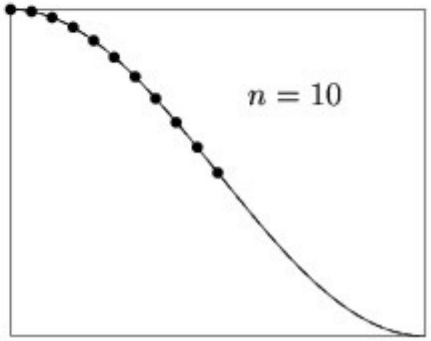




Convergence de l'interpolation polynomiale de cos(x)

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



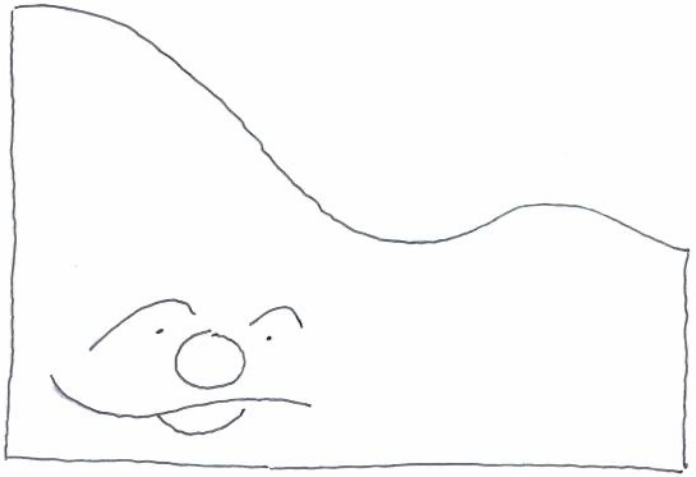


Définition 1.3.

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 :*

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

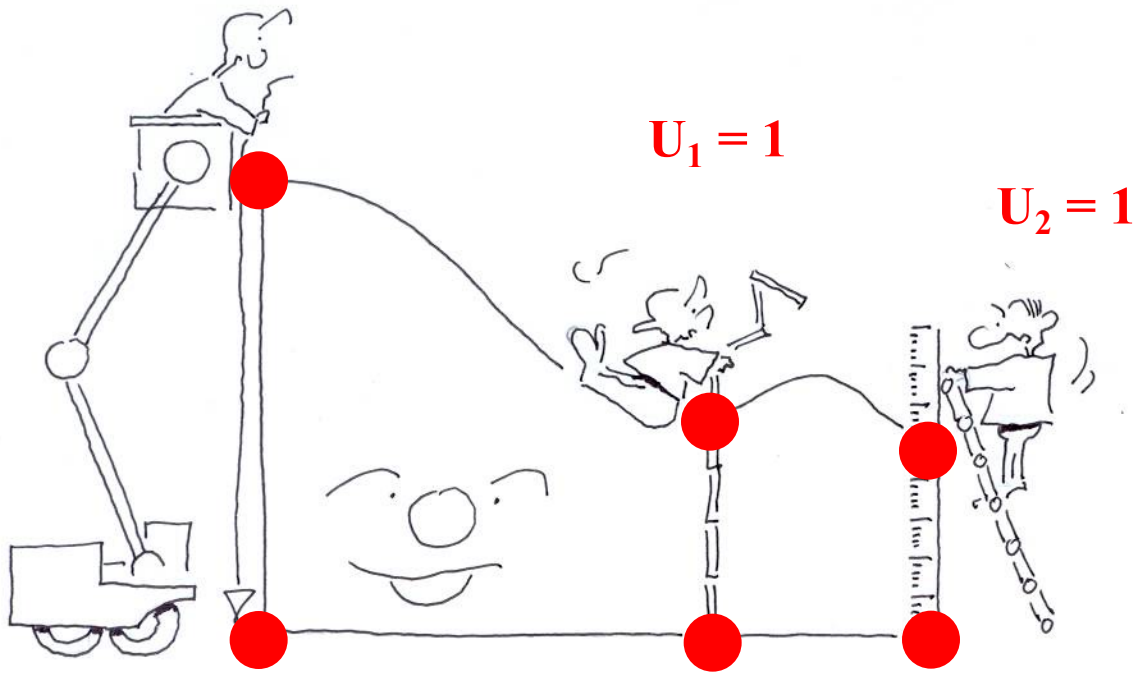
Une courbe….



Comment la représenter sur un ordinateur ?

Prise de mesures

$U_0 = 2$



$U_1 = 1$

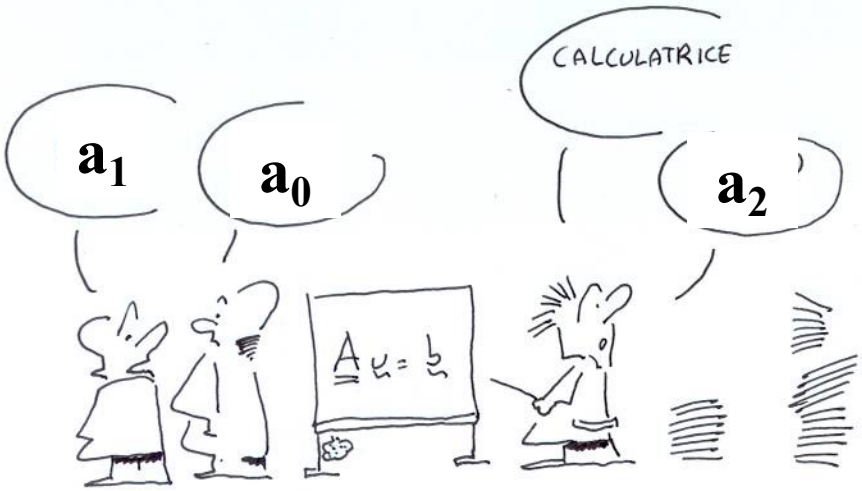
$U_2 = 1$

$X_0 = -1$

$X_1 = 1$

$X_2 = 2$

Utilisation des ressources informatiques facultaires

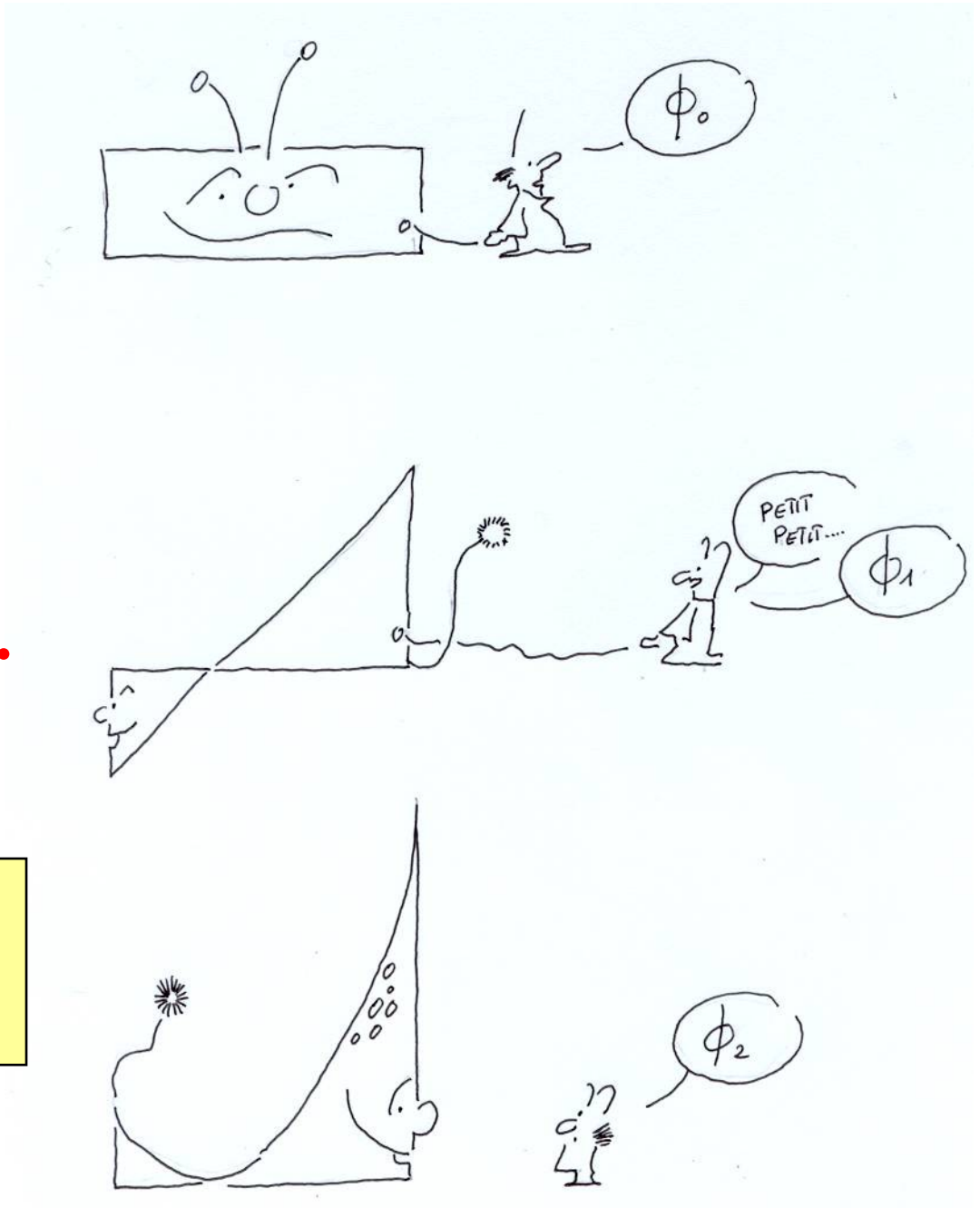


$a_1 = -1/2$

$a_2 = 1/6$

$a_0 = 4/3$

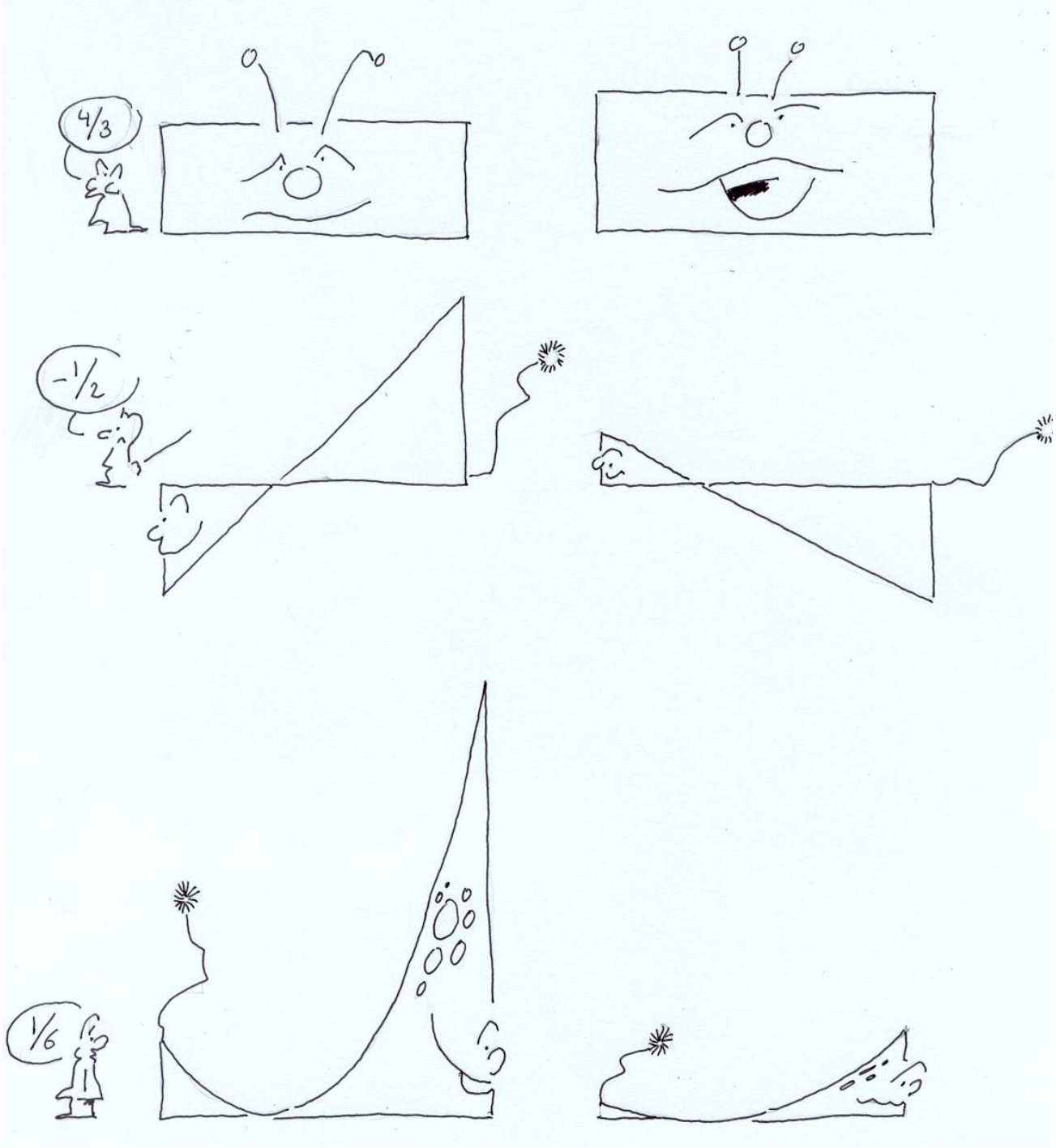
Allons chercher quelques monstres apprivoisés…



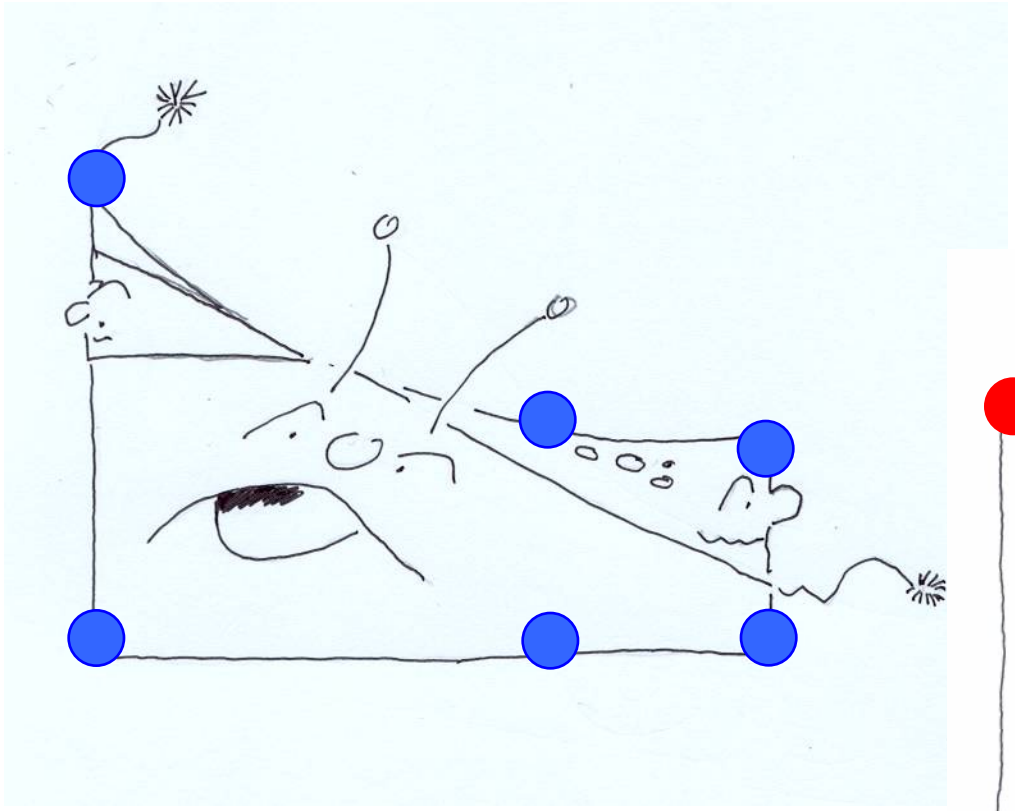
Les 3 fonctions de base d'espace discret de dimension 3.

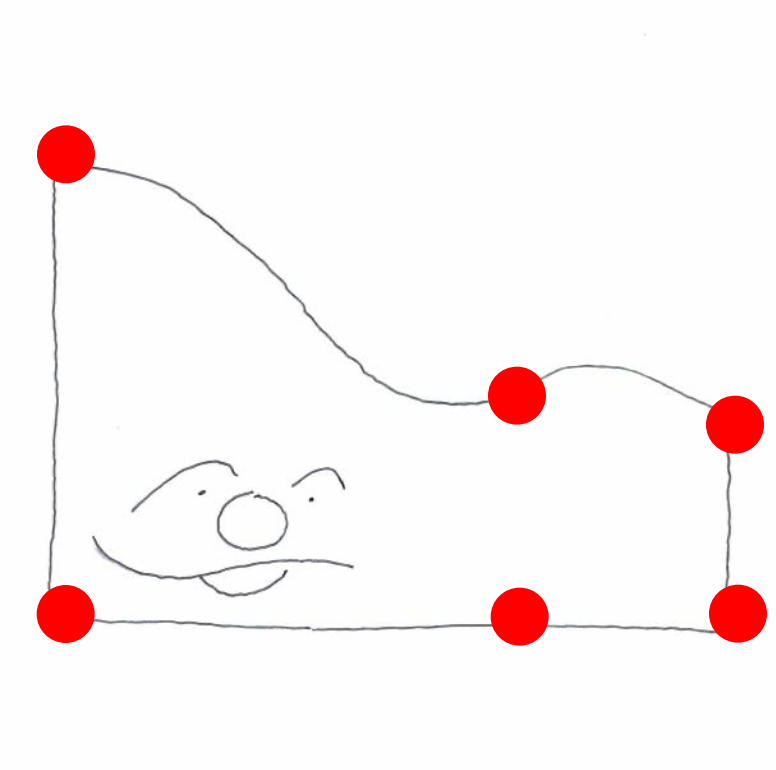
Base d'un espace vectoriel

Etes-vous un bon dresseur ?

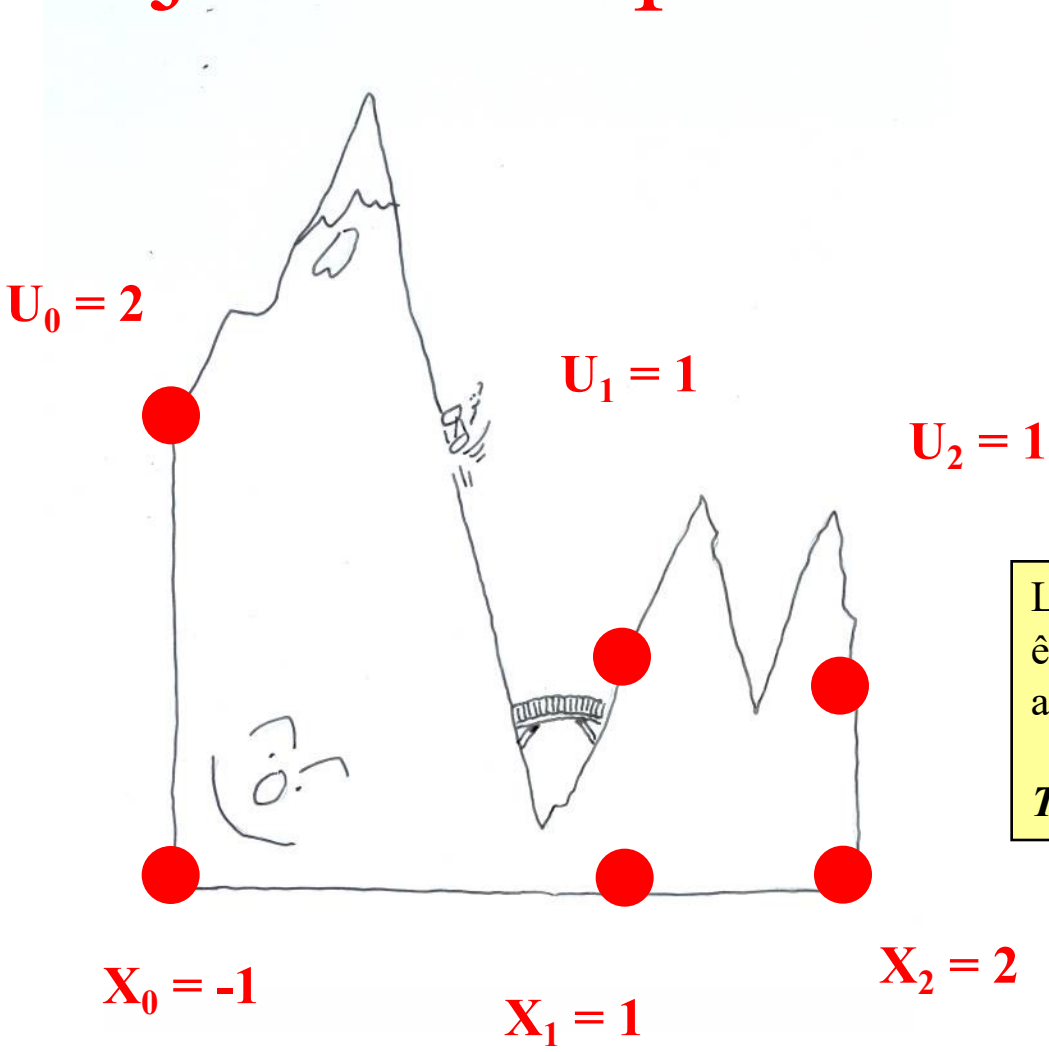


Et la phase critique…





Et si j'avais l'esprit montagnard…

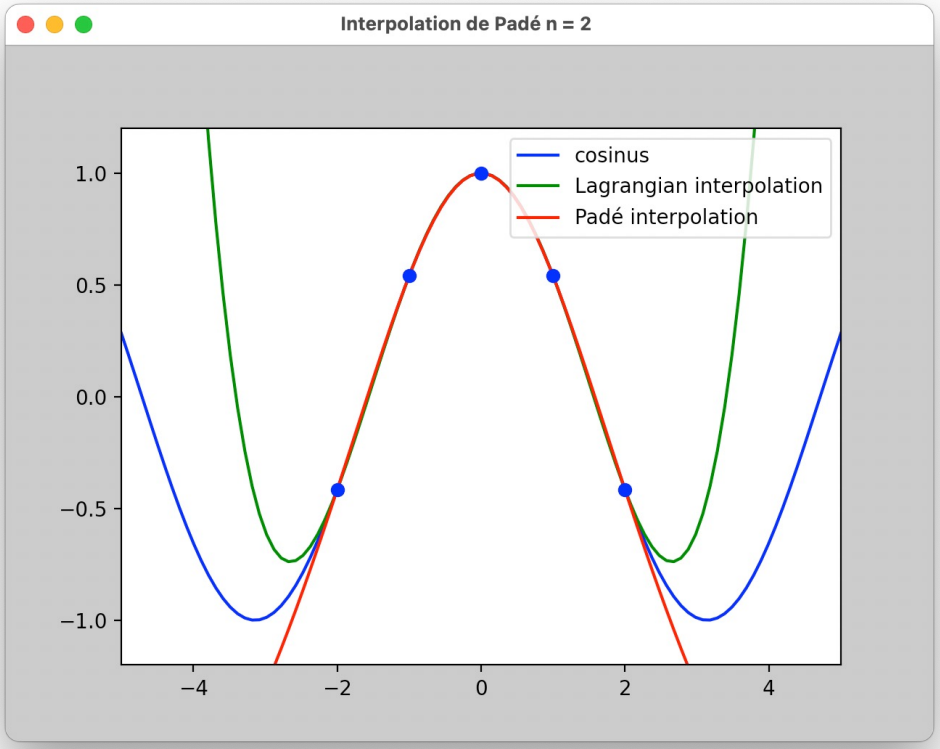


La prise de mesures ne peut pas être réalisée de manière arbitraire…

Théorie de l'échantillonnage

Interpolation de Padé

$$u(x) \approx u^h(x) = \frac{a_0 + a_1 x + a_2 x^2}{1 + a_3 x + a_4 x^2}$$



Homework 1

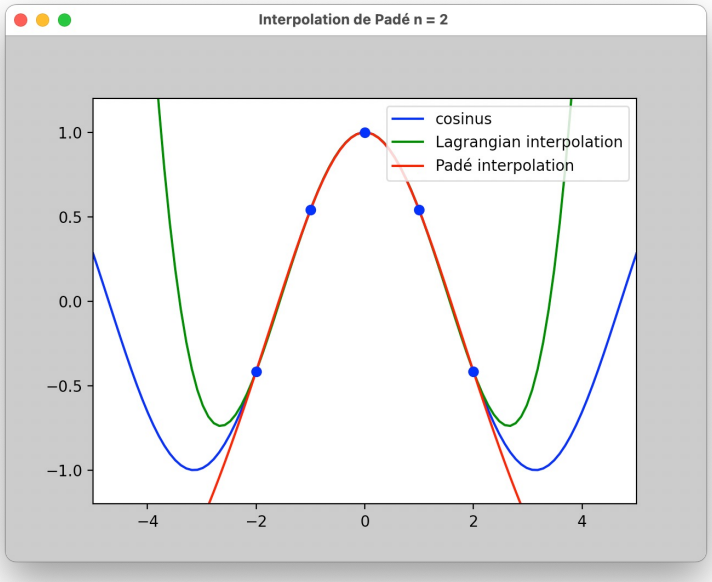
So easy !

$$\underbrace{U_i}_{u(X_i)} = \underbrace{\frac{a_0 + a_1 X_i + a_2 X_i^2}{1 + a_3 X_i + a_4 X_i^2}}_{u^h(X_i)}$$

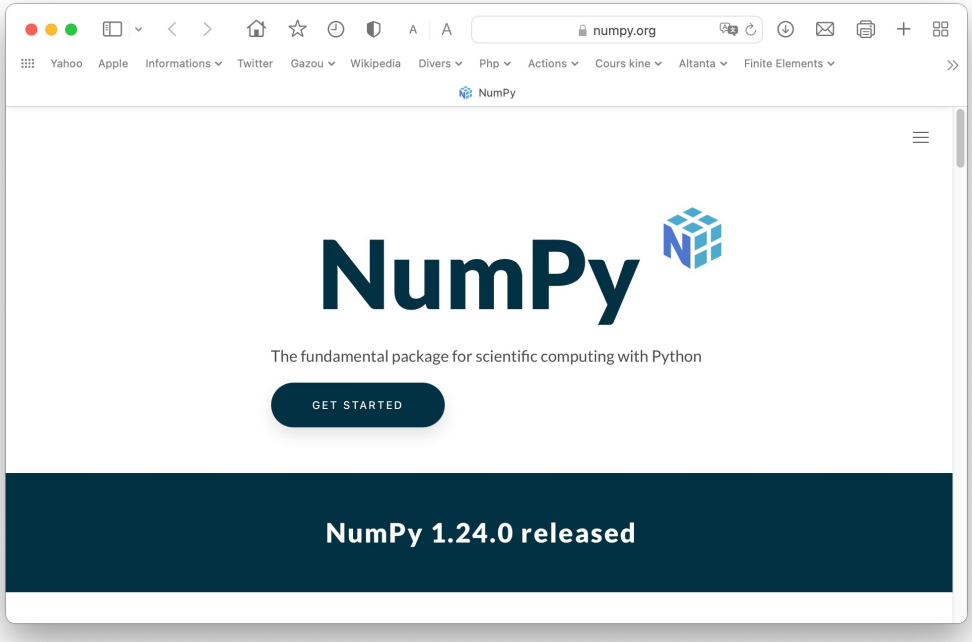
En espérant que le dénominateur ne vaille pas zéro :-)

$$U_i + a_3 U_i X_i + a_4 U_i X_i^2 = a_0 + a_1 X_i + a_2 X_i^2$$

$$a_0 + a_1 X_i + a_2 X_i^2 - a_3 U_i X_i - a_4 U_i X_i^2 = U_i$$



Utiliser numpy pour résoudre ce système !



$$\begin{bmatrix} 1 & X_0 & X_0^2 & -U_0X_0 & -U_0X_0^2 \\ 1 & X_1 & X_1^2 & -U_1X_1 & -U_1X_1^2 \\ 1 & X_1 & X_2^2 & -U_2X_2 & -U_2X_2^2 \\ 1 & X_3 & X_3^2 & -U_3X_3 & -U_3X_3^2 \\ 1 & X_4 & X_4^2 & -U_4X_4 & -U_4X_4^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

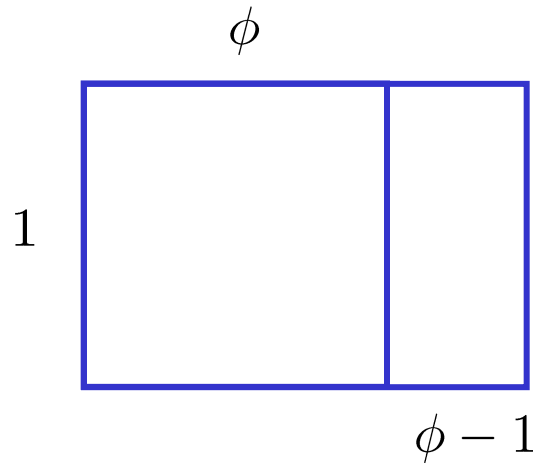
Le zéro problème

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

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
>>>
```

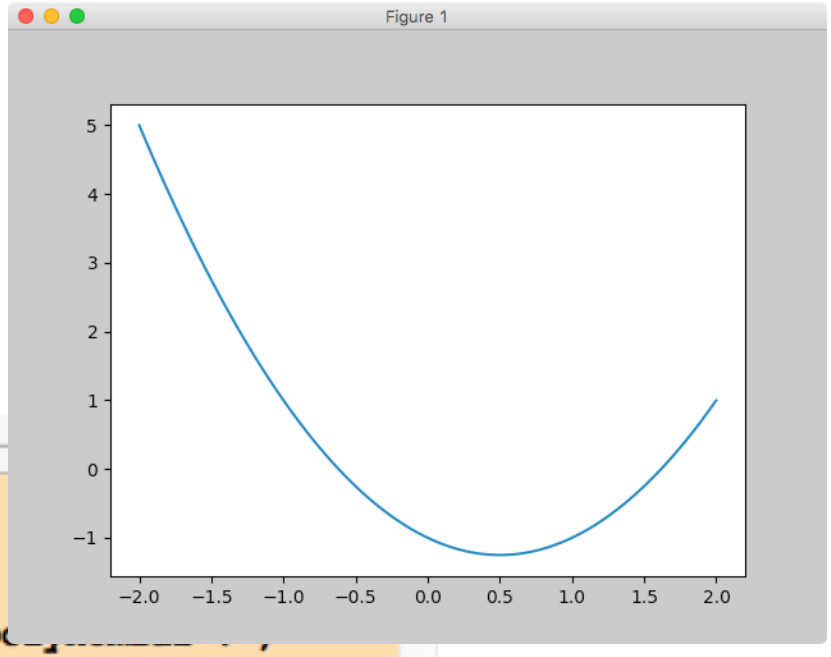
Polynômes avec Python

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

```
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
1.6180339887498949
 -1- Golden ratio number as roots of p
[ 1.61803399 -0.61803399]
```

Calcul symbolique : sympy

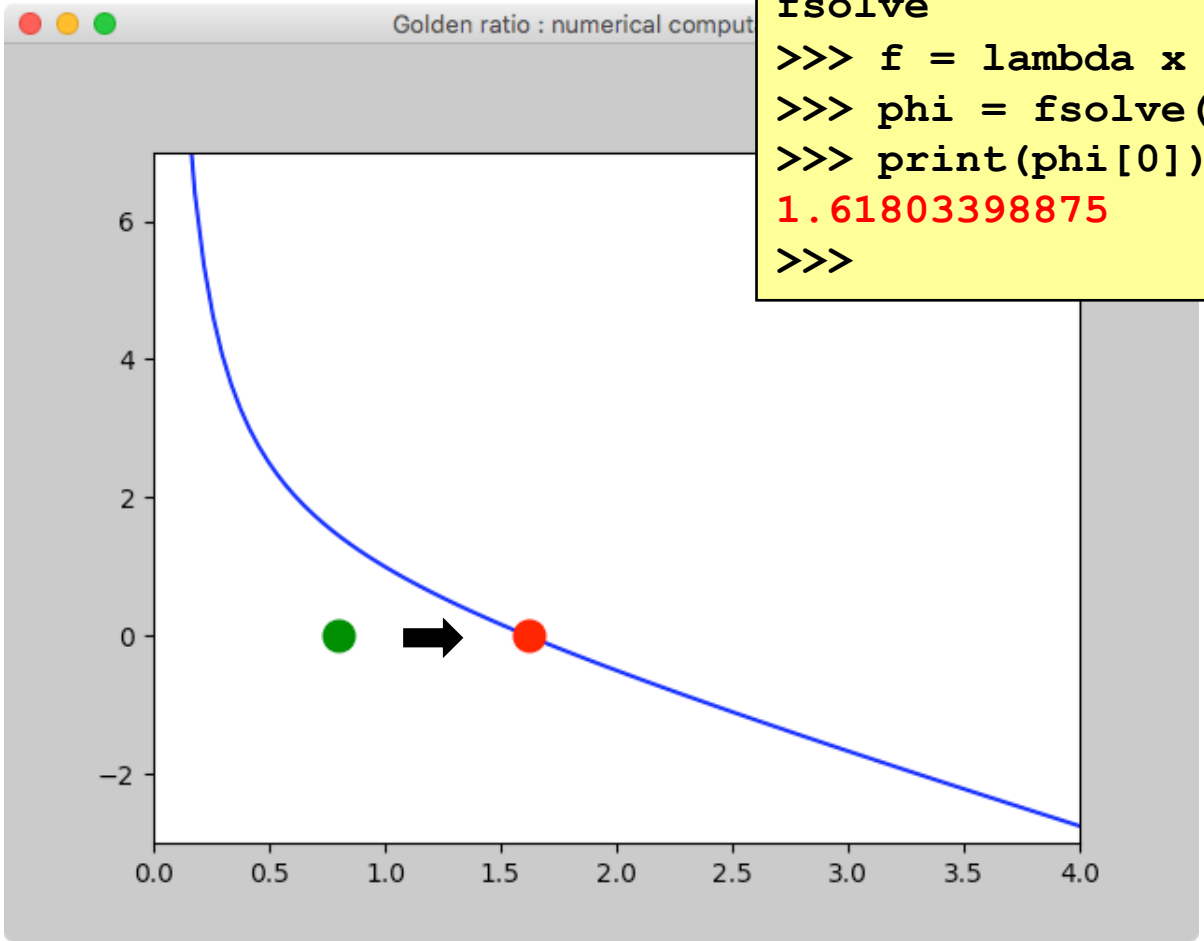
$$\phi^2 - \phi - 1 = 0$$

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

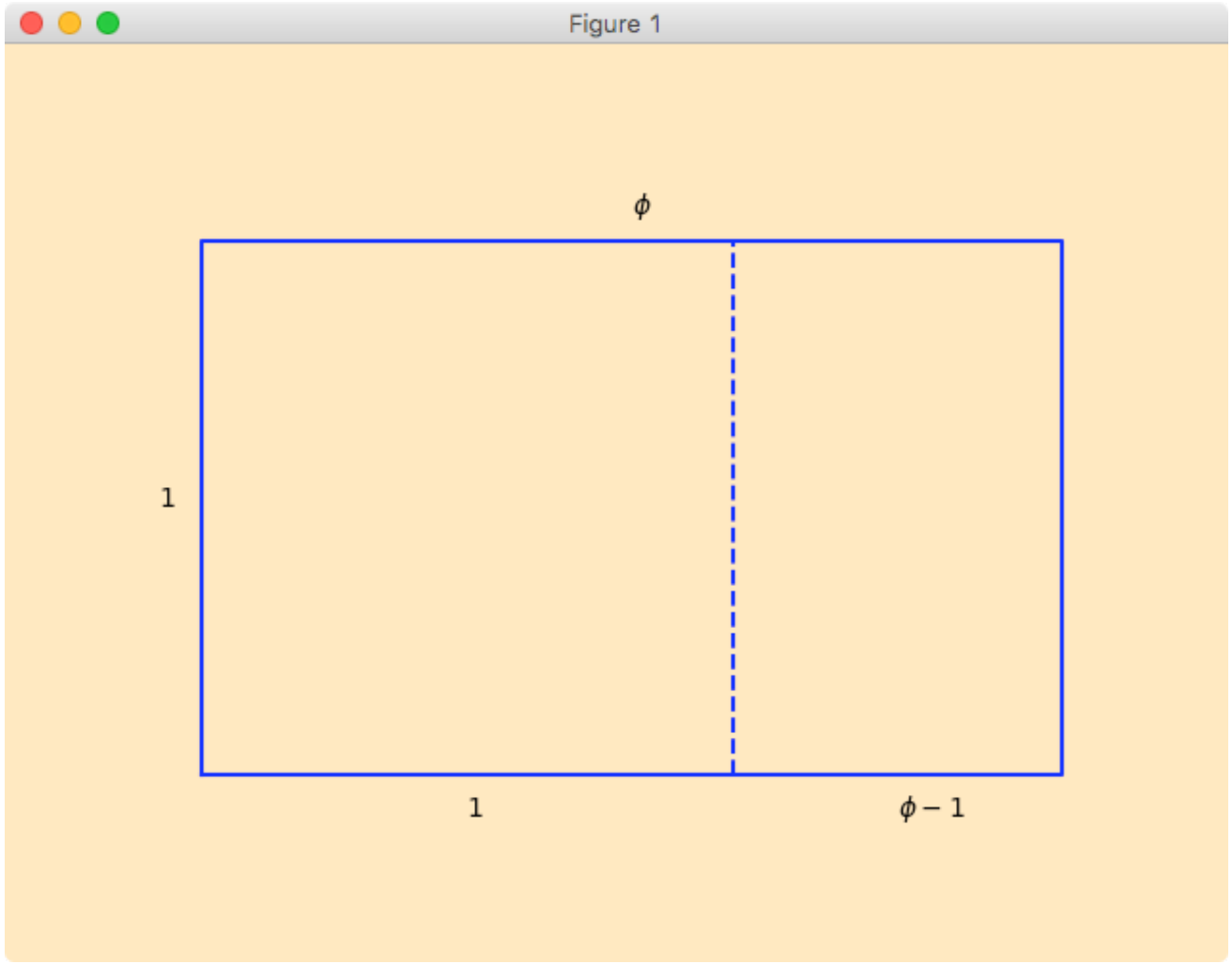
```
>>> from sympy import symbols,solve,evalf
>>> x = symbols('x')
>>> r = solve(1/x-x+1,x)
>>> print(r)
[1/2 + sqrt(5)/2, -sqrt(5)/2 + 1/2]
>>> phi = r[0]
>>> print(phi)
1/2 + sqrt(5)/2
>>> print(phi.evalf(50))
1.6180339887498948482045868343656381177203091798058
>>> print(phi.evalf())
1.61803398874989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

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
>>>
```



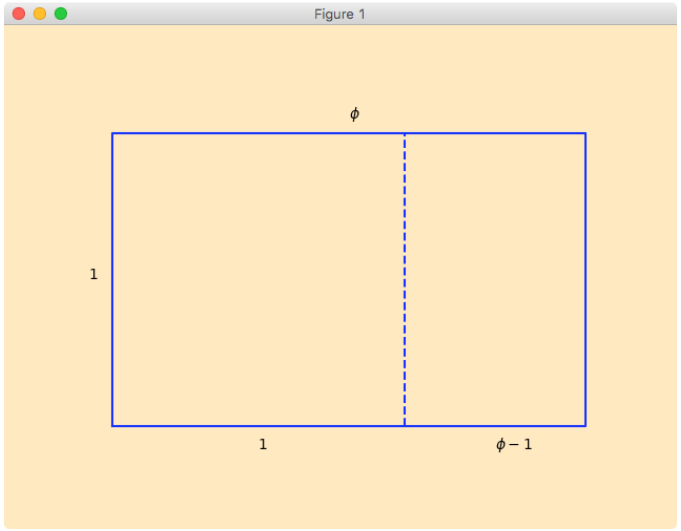
Faire des jolis dessins…



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…

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

$$\downarrow$$

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

$$\downarrow$$

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

```
>>> 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
```