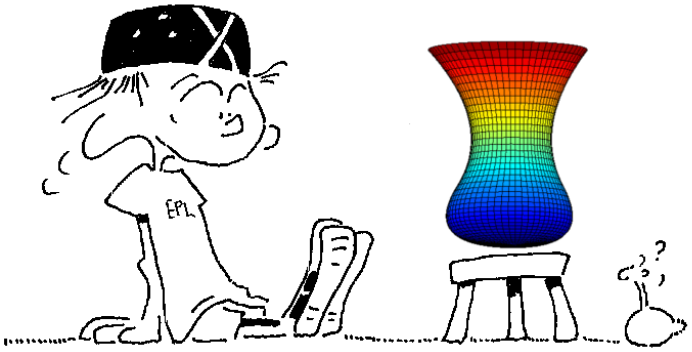
Ecole Polytechnique de Louvain



MATHEMATIQUES ET METHODES NUMERIQUES

...ou les aspects facétieux du calcul sur un ordinateur

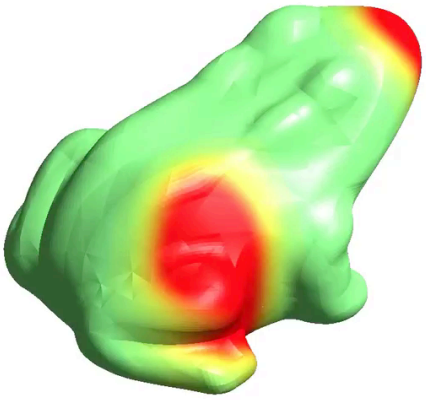
V. Legat

Copie des transparents pour le cours LEPL1104
Année académique 2022-2023 (version 2.4 : 04-01-2023)

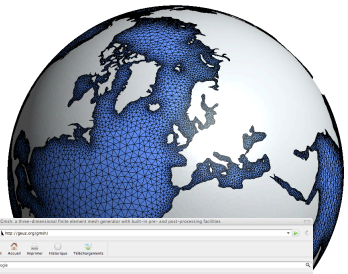
python™

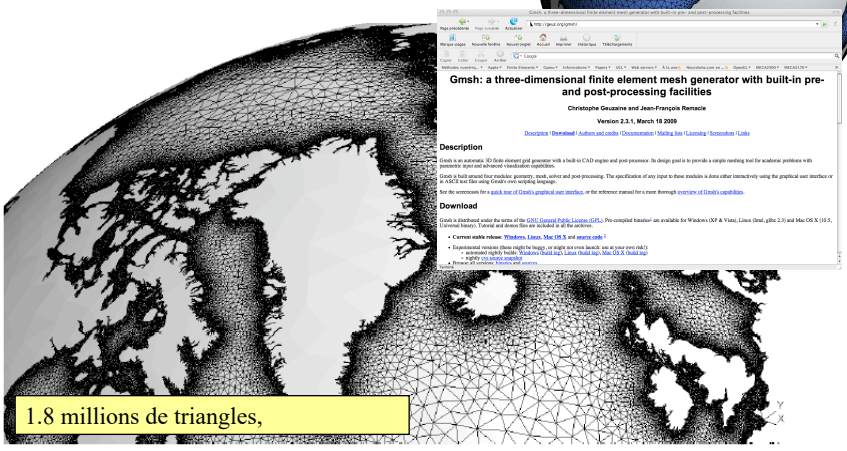
Avertissement : il s'agit uniquement de la copie des transparents utilisés l'année passée. Il est possible que certaines petites modifications soient effectuées : la dernière version sera mise sur le site web du cours après le cours.

Les méthodes numériques
A quoi cela sert-il ?



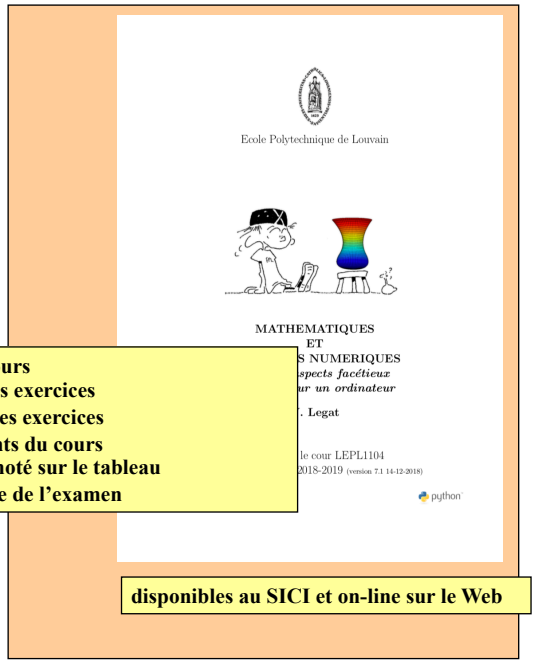
Second-generation Louvain-la-Neuve Ice-ocean Model





1.8 millions de triangles,

Notes de cours



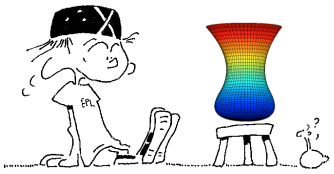
Notes de cours
Enoncés des exercices
Solutions des exercices
Transparents du cours
Ce qui est noté sur le tableau
= la matière de l'examen

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

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment approximer une fonction ?

Comment intégrer 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 ?

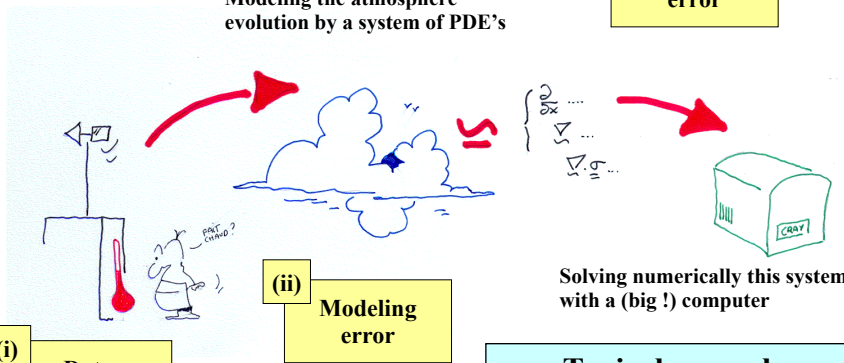
Comment résoudre numériquement une équation différentielle ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Weather forecasts are not always reliable…

(iii) **Computation error**

Modeling the atmosphere evolution by a system of PDE's



(ii) **Modeling error**

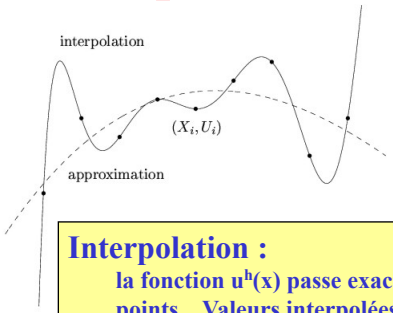
Solving numerically this system with a (big !) computer

(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

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

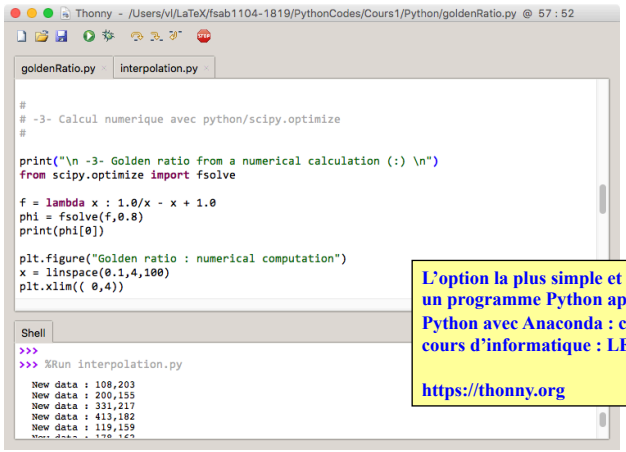
Né le 28 février 1735 à Paris
Décédé le 1er janvier 1796 à Paris

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

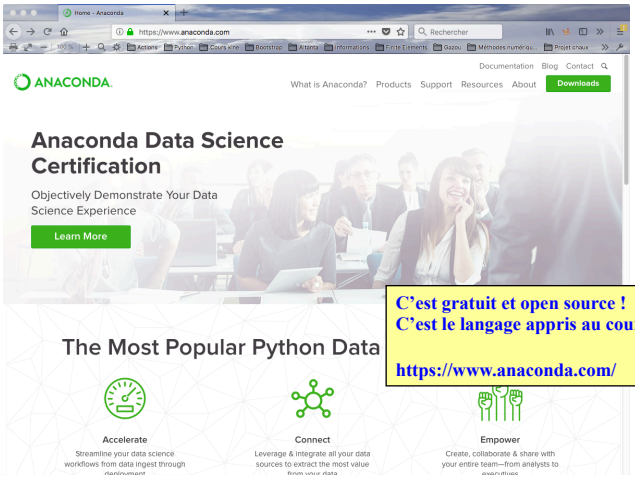
Un éditeur Python pour débutants !



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 ?

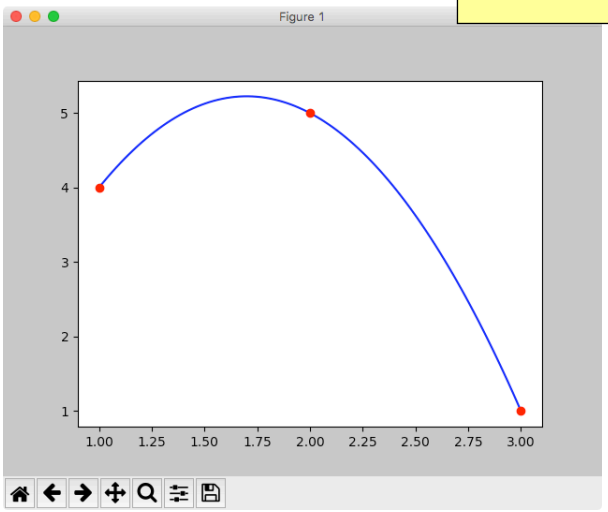


C'est gratuit et open source !
C'est le langage appris au cours d'informatique !

https://www.anaconda.com/

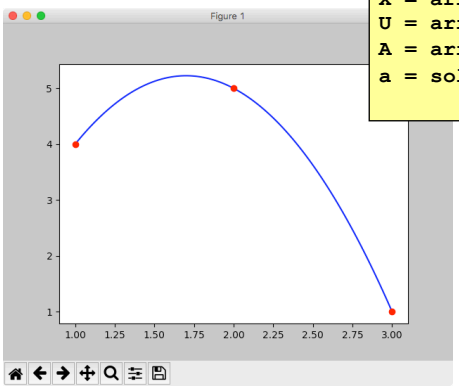
Faisons le calcul

```
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...

~~`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
```

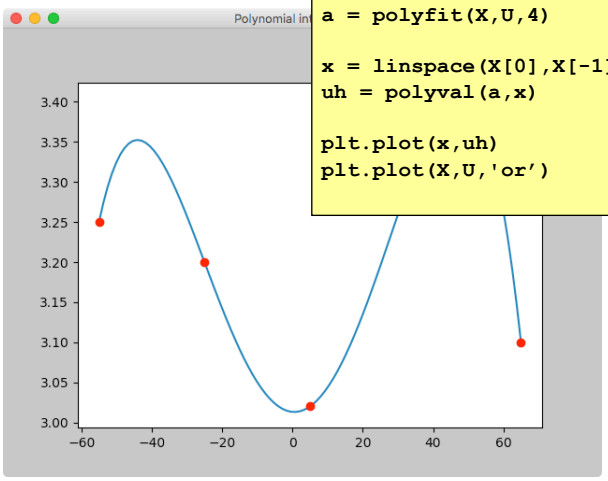
A better way, from both an execution time and numerical accuracy standpoint, is to use the matrix division operator

```
x = A\b
```

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

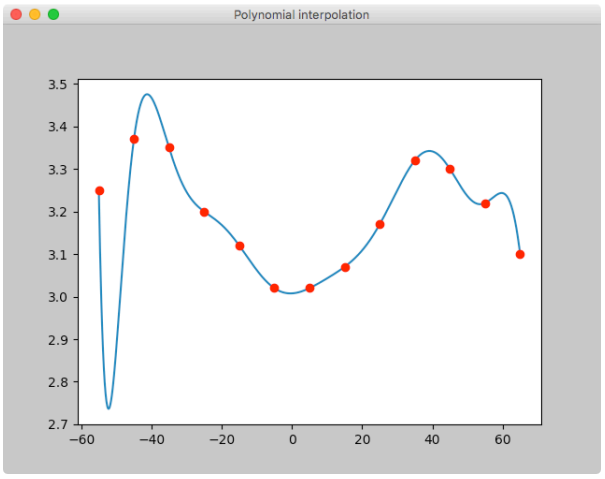
Exemple

```
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')
```

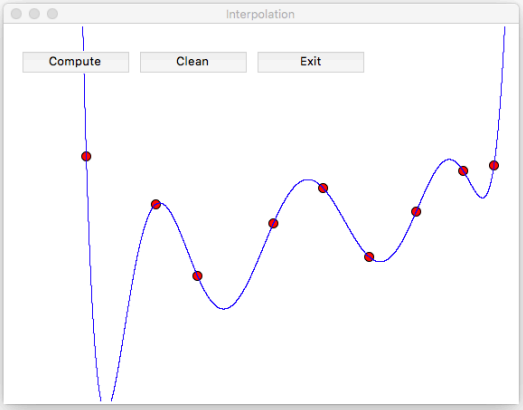


Davantage de données…

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

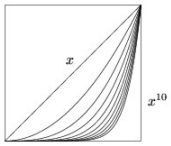


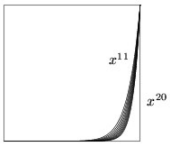
Le polynôme d'interpolation est un être bien turbulent…

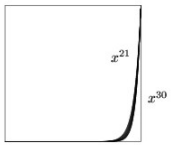


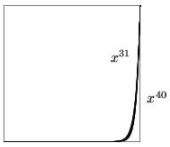
Les monômes

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









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

Matrice de Vandermonde…

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

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

C'est quoi un système mal conditionné ?

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

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

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

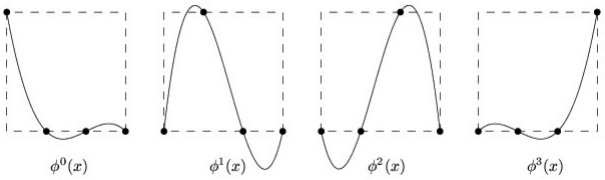
Solution perturbée : a=1.99999 et b=0.00001

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]))
```



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

Idée…

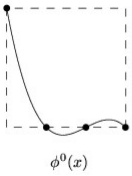
$$u^h(x) = \sum_{i=0}^{n} U_i \phi_i(x)$$

Les fonctions de base de Lagrange

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 !



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

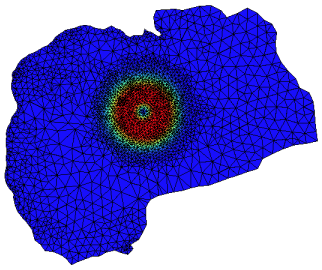
$$e^h(x) = u(x) - u^h(x)$$

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 :*

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

Est-ce utile ?



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

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

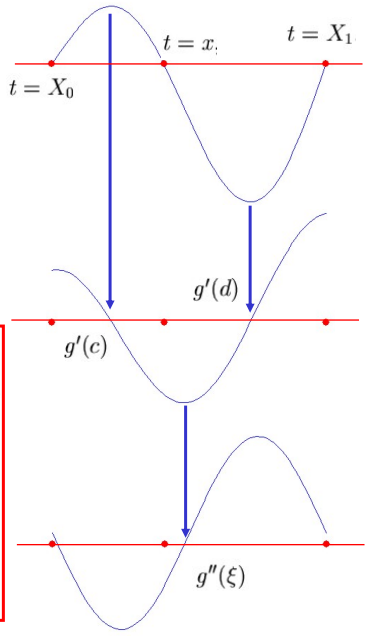
Intuitivement, on s'en serait un peu douté !

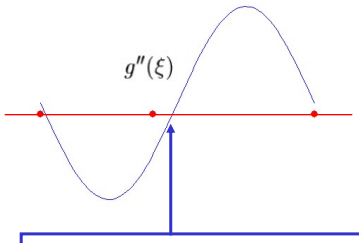
Nous allons démontrer uniquement le cas n=1
Ceci est le début de la démonstration :-)

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





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

□

Ceci est la fin de la démontration :-)

$$
\begin{aligned}
|e^h(x)| &\leq \frac{C_{n+1}}{(n+1)!}\underbrace{|(x-X_0)(x-X_1)\cdots(x-X_n)|}_{\leq n!\ h^{n+1}/4} \\
&\downarrow \quad \text{En définissant une nouvelle constante } C = \tfrac{n!C_{n+1}}{4(n+1)!} \\
|e^h(x)| &\leq Ch^{n+1} \\
&\downarrow \\
e^h(x) &= \mathcal{O}(h^{n+1})
\end{aligned}
$$

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

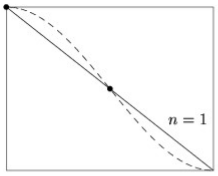
Définition 1.2. *Une approximation numérique dont le terme d'erreur est $\mathcal{O}(h^m)$ est dite d'ordre m.*

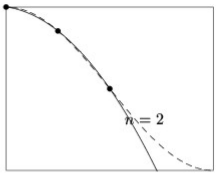
Polynôme de Taylor de degré n est souvent d'ordre n+1
Interpolation polynomiale par n+1 points équidistants est souvent d'ordre n+1

Signification intuitive de l'ordre d'une méthode numérique

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

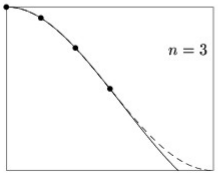
Convergence

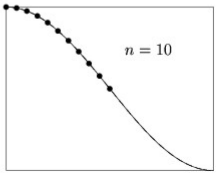




Convergence de l'interpolation polynomiale de cos(x)

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

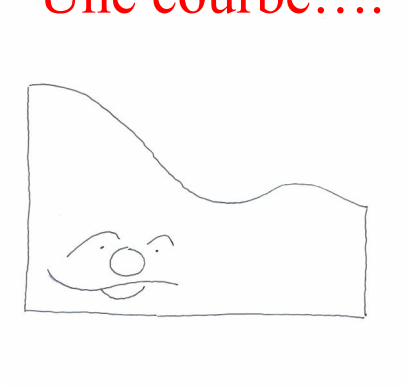




Définition 1.3. *Une interpolation est dite* convergente *si l'erreur d'interpolation tend vers zéro lorsque le nombre de degrés de liberté, c'est-à-dire n tend vers l'infini :*

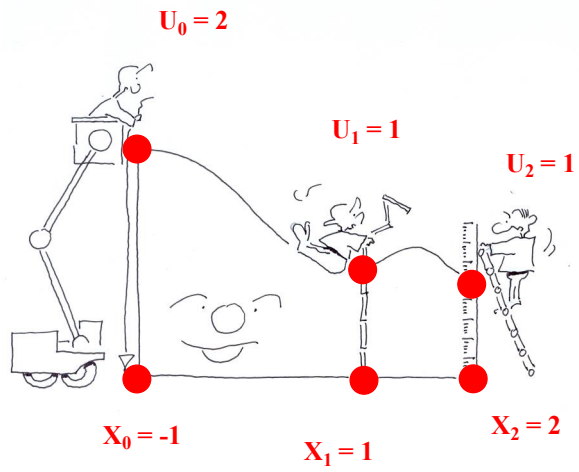
$$\lim_{n\to\infty} e^h(x) = 0 \quad \textit{pour } x \in [X_0, X_n].$$

Une courbe….

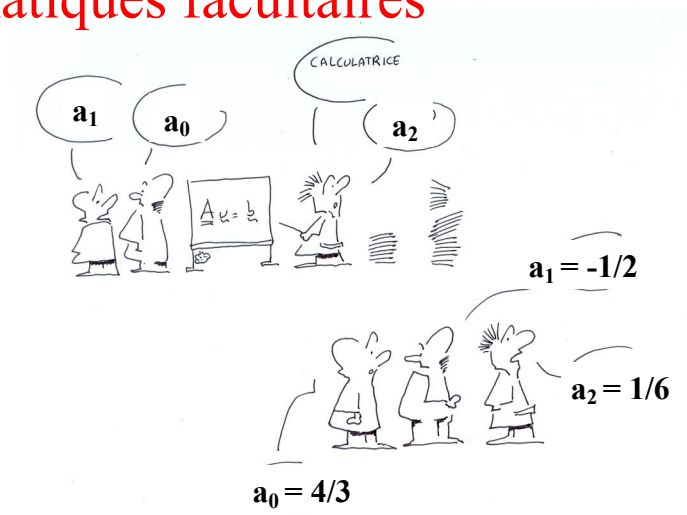


Comment la représenter sur un ordinateur ?

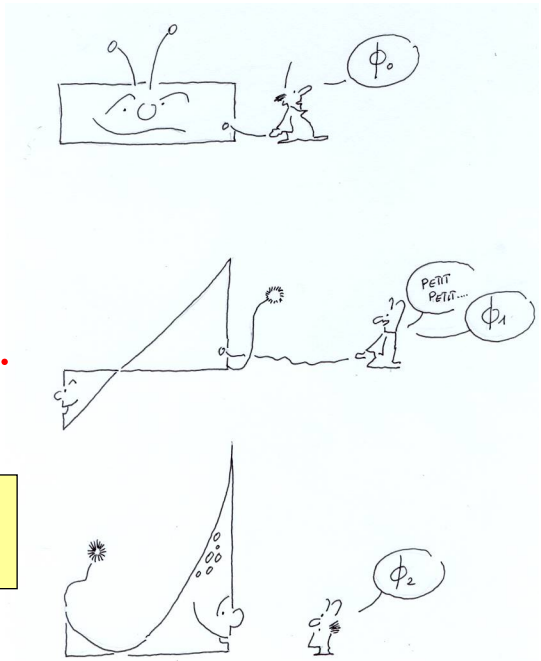
Prise de mesures



Utilisation des ressources informatiques facultaires



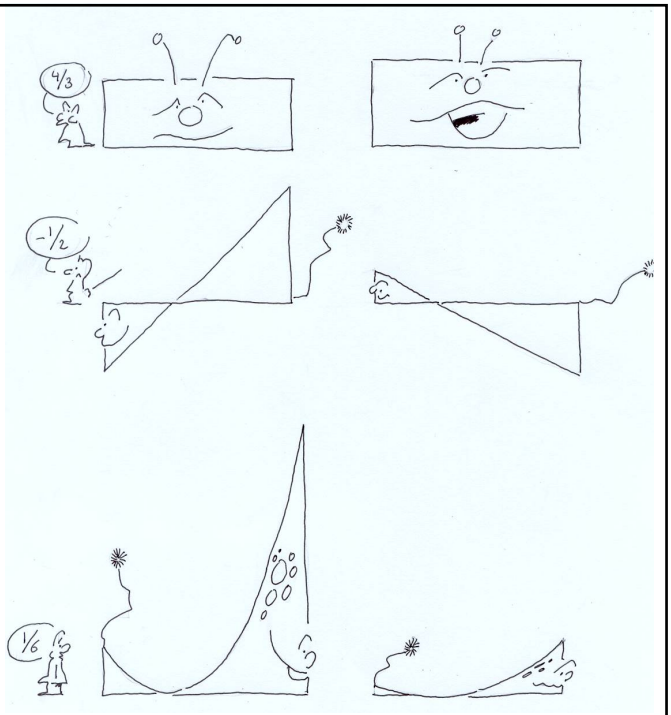
Allons chercher quelques monstres apprivoisés…



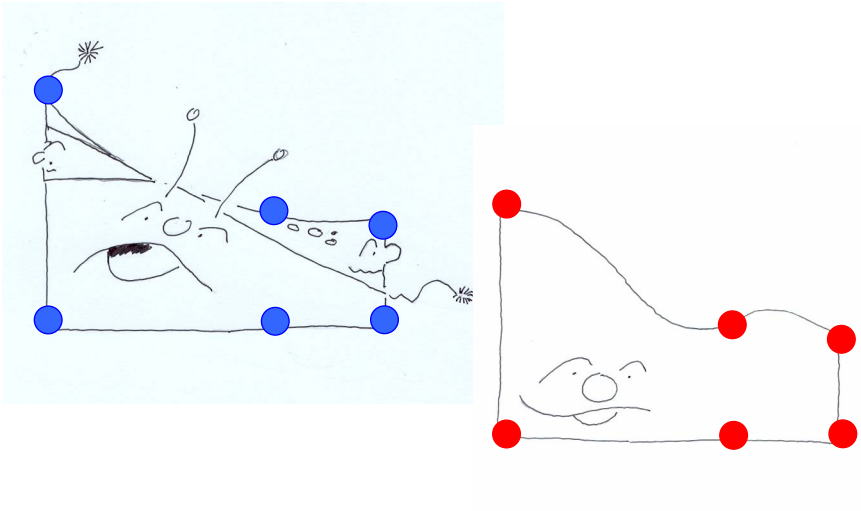
Les 3 fonctions de base d'espace discret de dimension 3.

Base d'un espace vectoriel

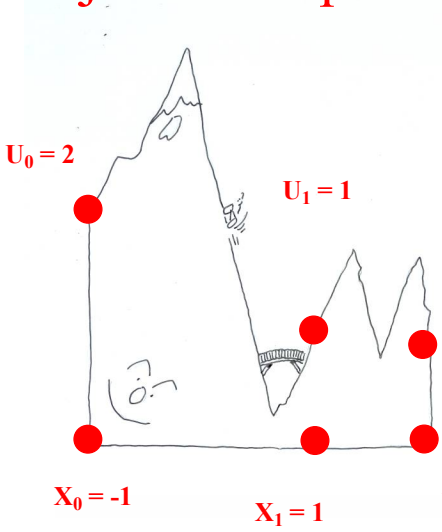
Etes-vous un bon dresseur ?



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

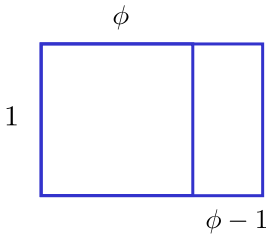
Le zéro problème

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

Nombre d'or

The Golden Ratio

```
from math import sqrt

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



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

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

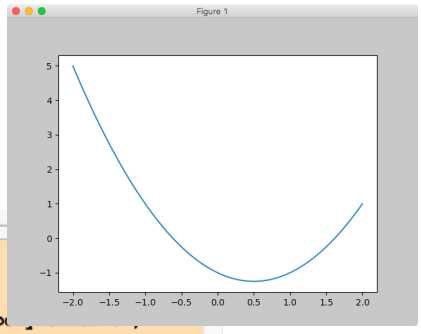
```
bash-3.2$ python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct  6 2017, 12:04:38)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from math import sqrt
>>> phi = (1.0 + sqrt(5.0)) / 2.0
>>> print(phi)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

Polynômes avec Python

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

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

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




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

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

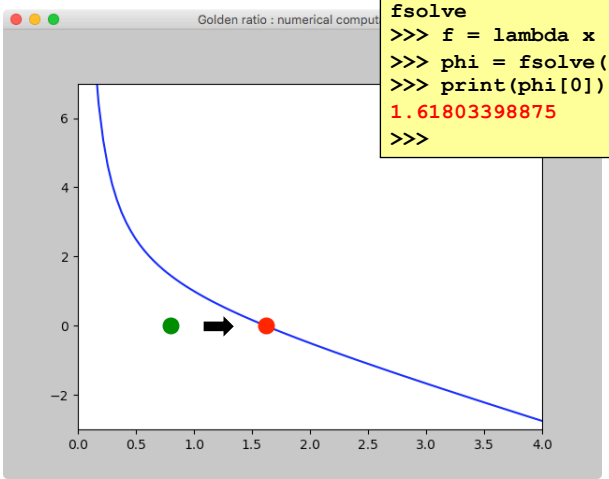
Calcul symbolique : sympy

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

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

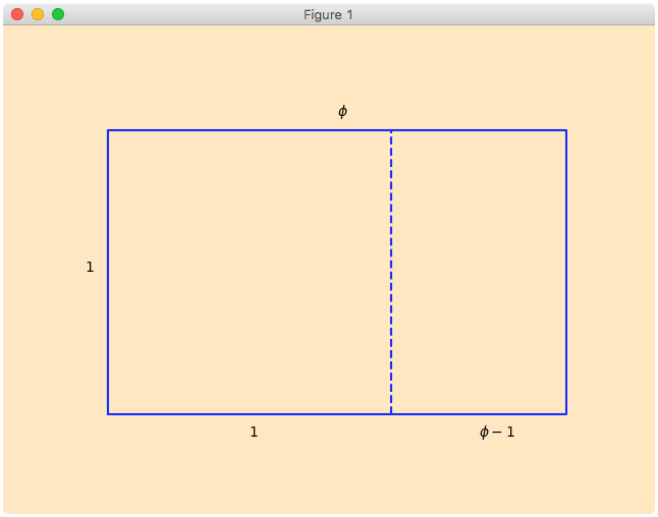
```
>>> from sympy import symbols,solve,evalf
>>> x = symbols('x')
>>> r = solve(1/x-x+1,x)
>>> print(r)
[1/2 + sqrt(5)/2, -sqrt(5)/2 + 1/2]
>>> phi = r[0]
>>> print(phi)
1/2 + sqrt(5)/2
>>> print(phi.evalf(50))
1.6180339887498948482045868343656381177203091798058
>>> print(phi.evalf())
1.61803398874989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

Résolution numérique : scipy

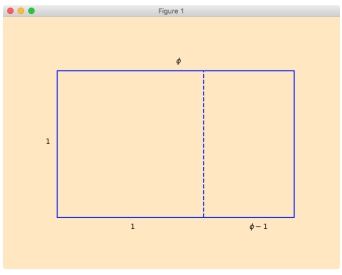



```
>>> from scipy.optimize import
fsolve
>>> f = lambda x : 1/x - (x-1)
>>> phi = fsolve(f,0.8)
>>> print(phi[0])
1.61803398875
>>>
```

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…

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

$$\downarrow$$

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

$$\downarrow$$

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

```
>>> 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…

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

$$\downarrow$$

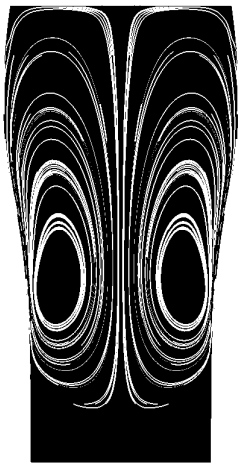
$$1 + \frac{1}{\phi} = \phi$$

$$\downarrow$$

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

```
>>> 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
```

A quoi servent les méthodes numériques ?



End of the Millenium Question

Do the Bubbles in a Glass of Guinness Beer Go Up or Down?

Since time immemorial, mankind has been troubled by natural phenomena that cannot be easily explained. Why do the bubbles in a glass of that venerable dark brew called Guinness appear to travel downward in the glass?

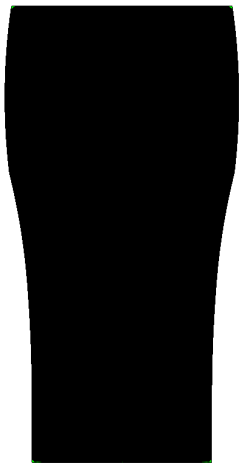
This goes against what we know about the physics of bubbles, or does it?

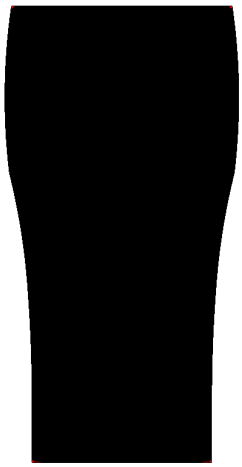
The answer to this most fundamental question has been locked away for centuries, resisting the direct assault of the most determined philosophers and theoreticians without compassion.

But, with the help of Computational Fluid Dynamics, we can provide a categorical and definitive answer: the bubbles go both up and down.

Fletcher et al. (1999)
CANCES-UNSW Australia
Fluent United States

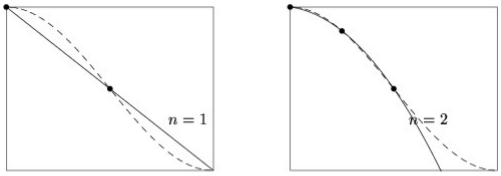
La réponse des scientifiques…





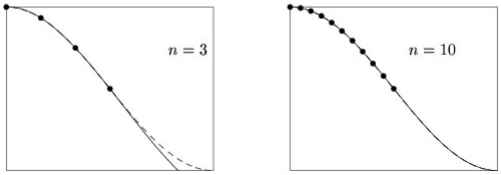
Bubble tracks show that the 1 mm bubbles move steadily upwards while the 60 micron bubbles are dragged downwards near the side of the glass.

Convergence



Convergence de l'interpolation polynomiale de cos(x)

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



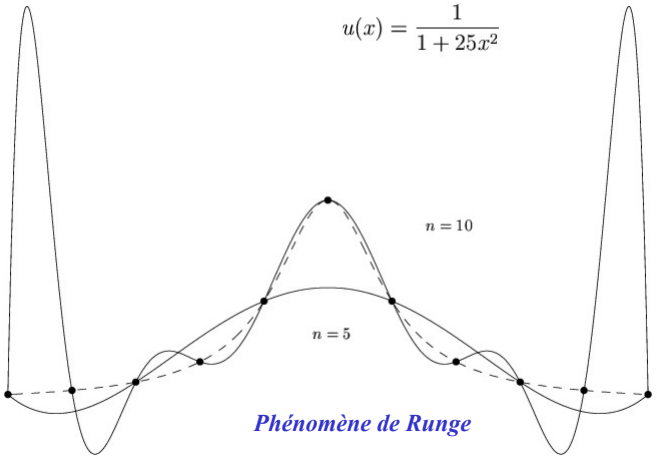
Définition 1.3.

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 :*

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

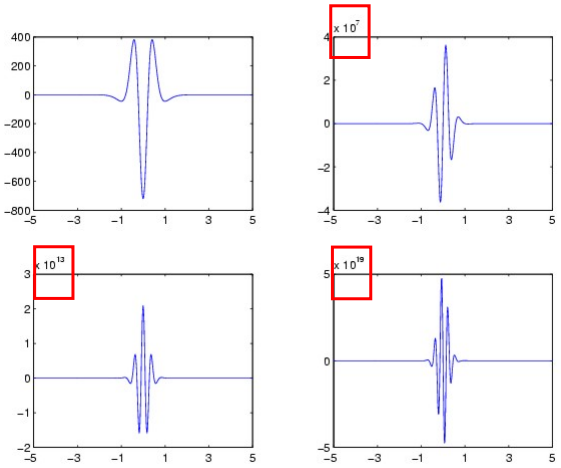
L'interpolation polynomiale, parfois cela ne converge pas…

$$u(x) = \frac{1}{1+25x^2}$$



Phénomène de Runge

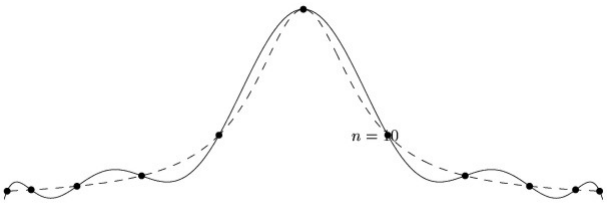
Why does it not work ?



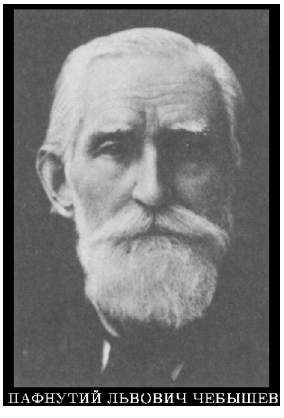
Dérivées d'ordre 6, 11, 16 et 21 de la fonction de Runge

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

Parfois, on peut sauver la mise…

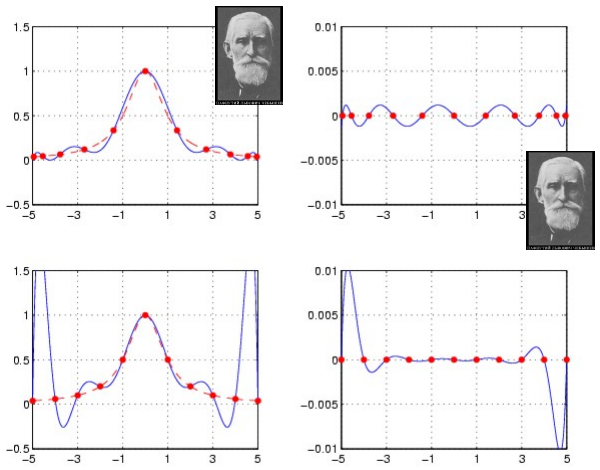


Abscisses de Chebyshev



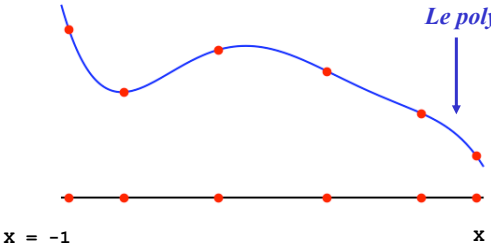
Pafnuty Lvovitch Chebyshev (1821-1894)

Why does it work ?



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

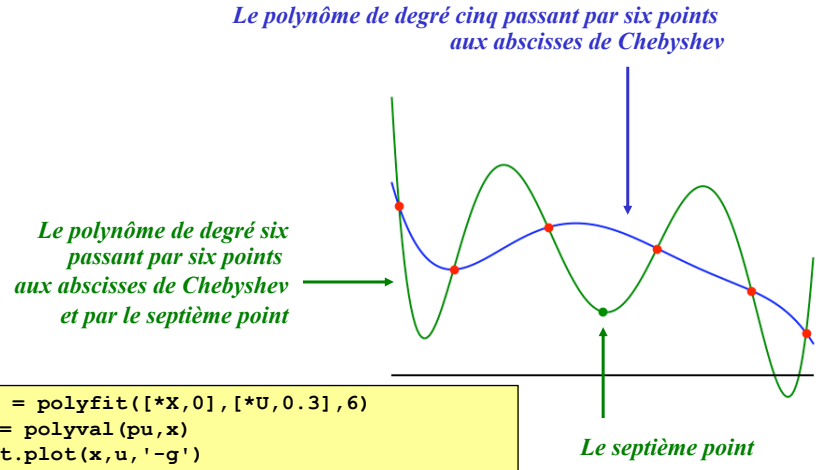
Six points aux abscisses de Chebyshev….



Le polynôme de degré cinq passant par six points aux abscisses de Chebyshev

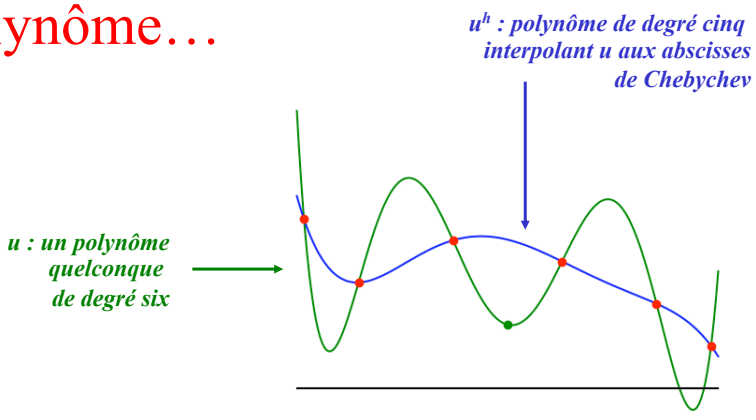
```
n = 5
X = cos(pi * (2*arange(0,n+1) + 1)/((n+1) * 2))
U = [0.2,0.4,0.6,0.7,0.5,0.8]

puh = polyfit(X,U,5)
x = linspace(-1,1,200)
uh = polyval(puh,x)
plt.plot(x,uh,'-b')
plt.plot(X,U,'or')
```

Ajoutons un septième point !

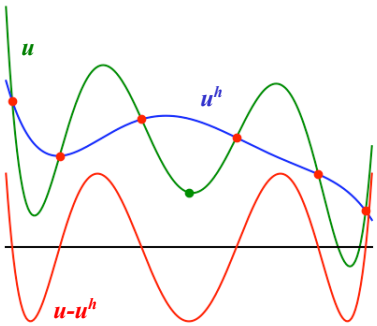



```
pu = polyfit([*X,0],[*U,0.3],6)
u = polyval(pu,x)
plt.plot(x,u,'-g')
plt.plot([0],[0.3],'og')
```

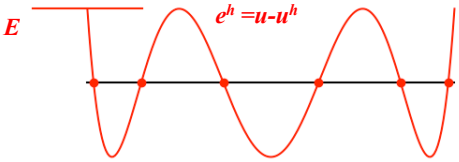
Interpolons un polynôme par un polynôme…



…c'est bête, je sais :-)

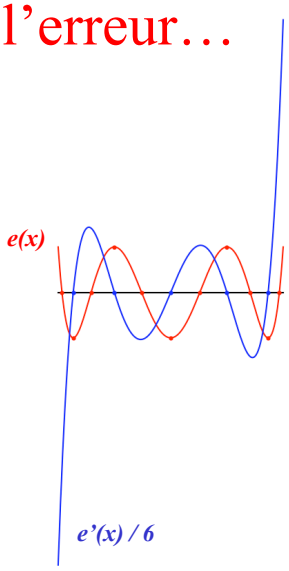



```
error = u - uh
E = error[0]
plt.plot(x,error,'-r');
```



Erreur d'interpolation

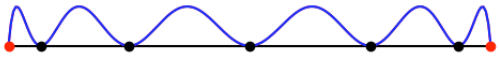
Dérivons et divisons l'erreur…




```
peh = pu - [0,*puh]
eh = polyval(peh, x)
plt.plot(x,eh,'-r')

pdeh = peh * [6,5,4,3,2,1,0]
pdeh = pdeh[0:-1]
deh = polyval(pdeh,x)/6
plt.plot(x,deh,'-b')
```

Et encore quelques petites manipulations…

$$(e'(x))^2\,(1-x^2) \;=\; (n+1)^2\,(E^2-e^2(x))$$



```
Xd = roots(pdeh)
Ed = polyval(peh,Xd)
E = Ed[0]

plt.plot(x,E**2-eh**2,'-r')
plt.plot(x,(1-x**2)*(deh**2),'-b')
plt.plot(Xd,zeros(size(Xd)),'ok')
plt.plot([-1,1],[0,0],'or')
```

Une solution analytique d'une équation différentielle ?

$$(e'(x))^2\,(1-x^2) \;=\; (n+1)^2\,(E^2-e^2(x))$$

$$\downarrow$$

$$\frac{\frac{e'(x)}{E}}{\sqrt{1-\left(\frac{e}{E}\right)^2}} \;=\; \pm(n+1)\frac{1}{\sqrt{1-x^2}}$$

$$\downarrow$$

```
>>> from sympy import *
>>> x = symbols('x')
>>> f = 1/sqrt(1-x**2)
>>> integrate(f)
asin(x)
```

Et si on dérive la primitive…

```
>>> from sympy import *
>>> x = symbols('x')
>>> f = 1/sqrt(1-x**2)
>>> integrate(f)
asin(x)
>>> diff(asin(x))
1/sqrt(-x**2 + 1)
>>> diff(acos(x))
-1/sqrt(-x**2 + 1)
```

Une petite solution analytique comme le faisaient les anciens…

$$(e'(x))^2 (1-x^2) = (n+1)^2 (E^2 - e^2(x))$$

$$\downarrow$$

$$\frac{\frac{e'(x)}{E}}{\sqrt{1-\left(\frac{e}{E}\right)^2}} = \pm(n+1)\frac{1}{\sqrt{1-x^2}}$$

$$\downarrow$$

$$\arccos\left(\frac{e(x)}{E}\right) = \pm\left((n+1)\arccos(x) + C\right)$$

$$\downarrow$$

En vertu de la parité du cosinus !

$$e(x) = E\cos\left((n+1)\arccos(x) + C\right)$$

$$\downarrow$$

En imposant que e(1) = E

$$e(x) = E\underbrace{\cos\left((n+1)\arccos(x)\right)}_{T_{n+1}(x)}$$

Polynôme de Chebyshev de degré n+1
Drôle d'expression pour un polynôme, non ?

Théorème 1.2.

Les polynômes de Chebyshev $T_{n+1}(x) = \cos((n+1)\arccos(x))$ *définis sur l'intervalle* $[-1,1]$ *satisfont la relation de récurrence*

$$T_{i+1}(x) = 2x\, T_i(x) - T_{i-1}(x), \quad i = 1, 2, 3, \ldots,$$

avec $T_0(x) = 1$ *et* $T_1(x) = x$.

Calcul des polynômes de Chebyshev : formule de récurrence

Démonstration : Définissons $\theta = \arccos(x)$ et écrivons :

$$\begin{aligned} T_{i+1}(x) &= \cos((i+1)\theta) \\ &= \cos(\theta)\cos(i\theta) - \sin(\theta)\sin(i\theta) \end{aligned}$$

$$\begin{aligned} T_{i-1}(x) &= \cos((i-1)\theta) \\ &= \cos(\theta)\cos(i\theta) + \sin(\theta)\sin(i\theta) \end{aligned}$$

$$\begin{aligned} T_{i+1}(x) + T_{i-1}(x) &= 2\cos(\theta)\cos(i\theta) \\ &= 2xT_i(x) \end{aligned}$$

□

Abscisses de Chebyshev

$$0 = \overbrace{\cos\left((n+1)\arccos(X_i)\right)}^{T_{n+1}(X_i)} \qquad i = 0, \ldots, n$$

$$\downarrow$$

$$\frac{\pi/2 + i\pi}{(n+1)} = \arccos(X_i) \qquad i = 0, \ldots, n$$

$$\downarrow$$

$$\cos\left(\frac{(2i+1)\pi}{2(n+1)}\right) = X_i \qquad i = 0, \ldots, n$$

Interpolation polynomiale : bilan

- **Pour une fonction u(x) très régulière : fonction cosinus**

  Convergence de l'interpolation polynomiale

- **Pour une fonction u(x) suffisamment régulière : fonction de Runge**

  Divergence pour des abscisses équidistantes
  Convergence pour les abscisses de Chebyshev

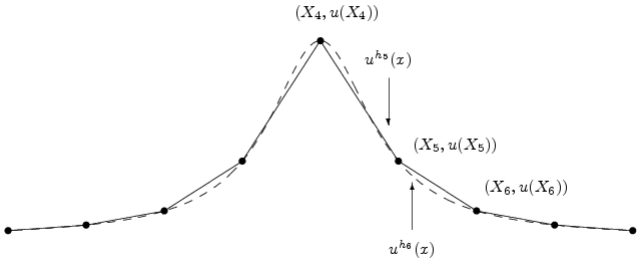
- **Pour une fonction u(x) peu régulière : fonction échelon**

  Divergence !
  Eviter l'interpolation polynomiale de degré élevé

Idée :

Utiliser une interpolation par morceaux composée des polynômes de degré bas !

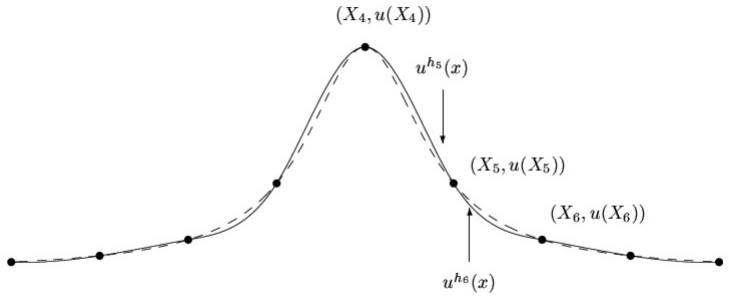
Interpolation linéaire par morceaux



Interpolation par splines cubiques

$$u^{h_i}(x) = a_i + b_i x + c_i x^2 + d_i x^3 \qquad i = 1, 2, \ldots, n$$

4n coefficients inconnus



Comment trouver les coefficients ?

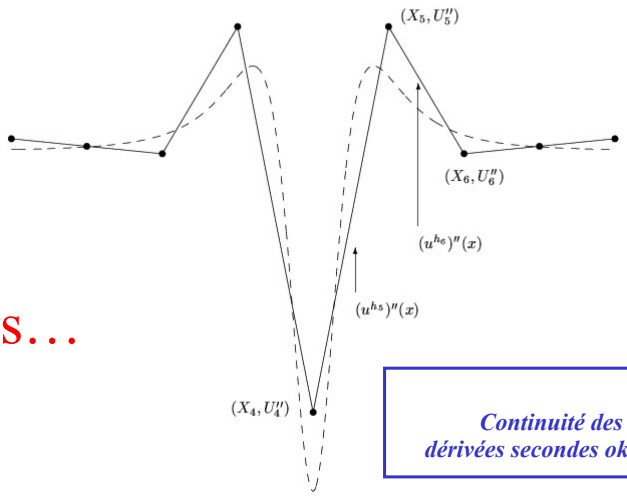
$$\begin{aligned} u^{h_1}(X_0) &= U_0 \\ u^{h_n}(X_n) &= U_n \end{aligned}$$

$$\begin{aligned} u^{h_i}(X_i) &= U_i \qquad i = 1, \ldots, n-1 \\ u^{h_{i+1}}(X_i) &= U_i \qquad i = 1, \ldots, n-1 \end{aligned}$$

$$\begin{aligned} (u^{h_i})'(X_i) &= (u^{h_{i+1}})'(X_i) \qquad i = 1, \ldots, n-1 \\ (u^{h_i})''(X_i) &= (u^{h_{i+1}})''(X_i) \qquad i = 1, \ldots, n-1 \end{aligned}$$

4n-2 conditions

$$(u^{h_i})'' = U''_{i-1}\frac{(x - X_i)}{(X_{i-1} - X_i)} + U''_i\frac{(x - X_{i-1})}{(X_i - X_{i-1})}$$



Ecrivons…

Continuité des dérivées secondes ok

Intégrons…

$$(u^{h_i})'' = U''_{i-1}\frac{(X_i - x)}{h_i} + U''_i\frac{(x - X_{i-1})}{h_i}$$

En intégrant deux fois,

$$(u^{h_i}) = U''_{i-1}\frac{(X_i - x)^3}{6h_i} + U''_i\frac{(x - X_{i-1})^3}{6h_i} + A_i\frac{(X_i - x)}{h_i} + B_i\frac{(x - X_{i-1})}{h_i}$$

$$U_{i-1} = U''_{i-1}\frac{h_i^3}{6h_i} + A_i\frac{h_i}{h_i} \quad \text{et} \quad U_i = U''_i\frac{h_i^3}{6h_i} + B_i\frac{h_i}{h_i}$$

$$A_i = U_{i-1} - \frac{U''_{i-1}\,h_i^2}{6} \qquad B_i = U_i - \frac{U''_i\,h_i^2}{6}$$

Continuité de la fonction ok

Calculons…

Continuité des dérivées premières ok

$$(u^{h_i})'(X_i) = (u^{h_{i+1}})'(X_i)$$

$$\frac{U_i'' h_i}{2} + \frac{(U_i - U_{i-1})}{h_i} - \frac{(U_i'' - U_{i-1}'')h_i}{6} = -\frac{U_i'' h_{i+1}}{2} + \frac{(U_{i+1} - U_i)}{h_{i+1}} - \frac{(U_{i+1}'' - U_i'')h_{i+1}}{6}$$

$$\frac{(2U_i'' + U_{i-1}'')h_i}{6} + \frac{(U_i - U_{i-1})}{h_i} = \frac{(U_{i+1} - U_i)}{h_{i+1}} - \frac{(U_{i+1}'' + 2U_i'')h_{i+1}}{6}$$

$$\frac{h_i}{6} U_{i-1}'' + \frac{2(h_i + h_{i+1})}{6} U_i'' + \frac{h_{i+1}}{6} U_{i+1}'' = \frac{(U_{i+1} - U_i)}{h_{i+1}} - \frac{(U_i - U_{i-1})}{h_i}$$

$$i = 1, \ldots, n-1$$

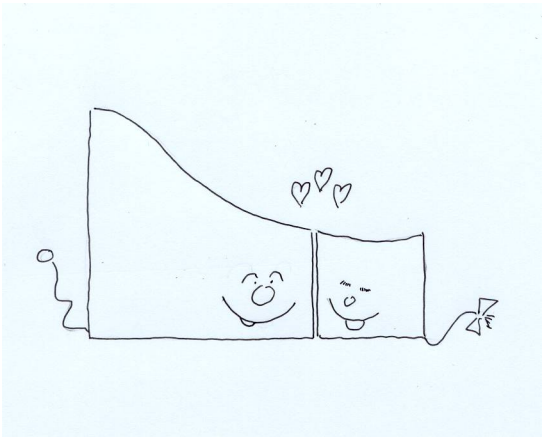
Abscisses équidistantes

2 conditions supplémentaires
Courbe spline naturelle

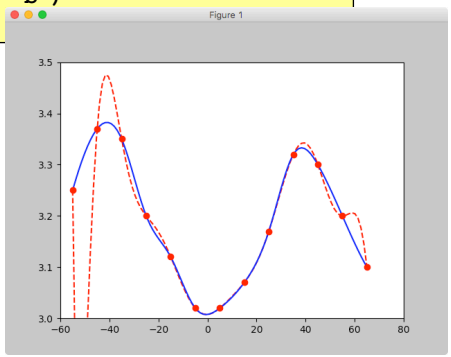
$$U_0'' = 0 \quad \text{et} \quad U_n'' = 0$$

$$\frac{h^2}{6} \begin{bmatrix} 1 & 0 & & & & & & \\ 1 & 4 & 1 & & & & & \\ & 1 & 4 & 1 & & & & \\ & & 1 & 4 & 1 & & & \\ & & & 1 & 4 & 1 & & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & 1 & 4 & 1 & \\ & & & & & 1 & 4 & 1 \\ & & & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} U_0'' \\ U_1'' \\ U_2'' \\ U_3'' \\ U_4'' \\ \vdots \\ U_{n-2}'' \\ U_{n-1}'' \\ U_n'' \end{bmatrix} = \begin{bmatrix} 0 \\ U_0 - 2U_1 + U_2 \\ U_1 - 2U_2 + U_3 \\ U_2 - 2U_3 + U_4 \\ U_3 - 2U_4 + U_5 \\ \vdots \\ \vdots \\ U_{n-2} - 2U_{n-1} + U_n \\ 0 \end{bmatrix}$$

Ou de manière plus poétique…

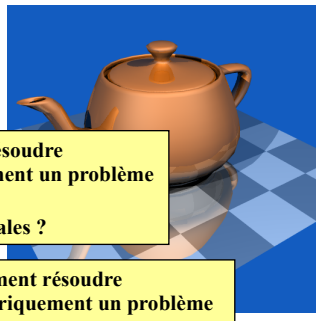


```
from numpy import *
from scipy.interpolate import CubicSpline as spline
from matplotlib import pyplot as plt

X = arange(-55,70,10)
U = [3.25, 3.37, 3.35, 3.20, 3.12, 3.02, 3.02,
          3.07, 3.17, 3.32, 3.30, 3.20, 3.10]
x = linspace(X[0],X[-1],100)
uhLag = polyval(polyfit(X,U,len(X)-1),x)
uhSpl = spline(X,U)

plt.plot(x,uhLag,'--r',x,uhSpl(x),'-b')
plt.plot(X,U,'or')
```



Exemple

Plan des cours de méthodes numériques



Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement un problème aux valeurs initiales ?

Comment résoudre numériquement un problème aux conditions frontières ?

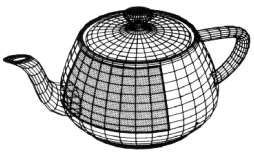
Et les équations non linéaires ?

Et les méthodes itératives ?

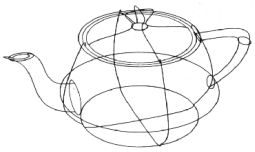
Comment résoudre numériquement une équation aux dérivées partielles ?

The Utah Teapot

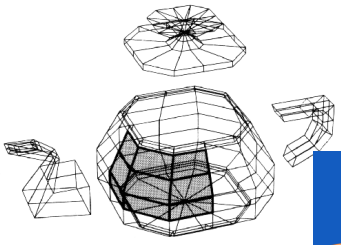
Martin Nevell (1975)



Single shaded patch



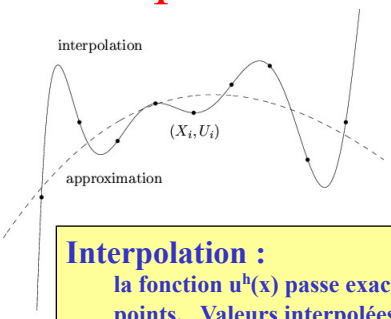
Patch edges



Control points

32 patches

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.

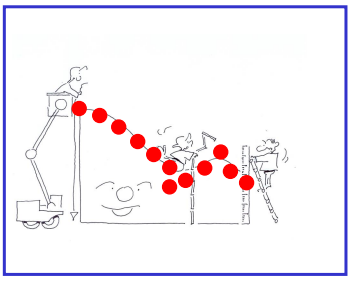
Approximation :

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

Beaucoup de données…

m = 11

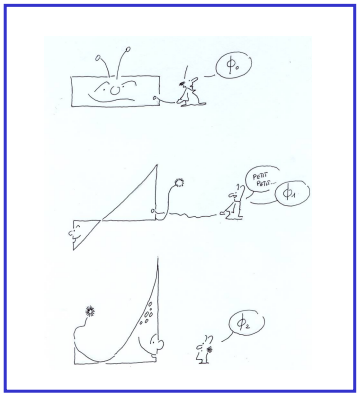
$$(X_i, U_i), \quad i = 0, 1, 2, \ldots, m.$$



$$n < m$$

n = 2

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



…peu de paramètres !

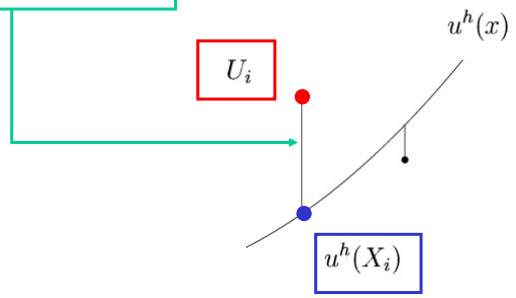
La fonction $u^h(x)$ ne passe pas par les points, mais s'en rapproche selon un critère à définir…

$$u^h(X_i) \approx u(X_i)$$

$$\begin{bmatrix} \phi_0(X_0) & \phi_1(X_0) & \dots & \phi_n(X_0) \\ \phi_0(X_1) & \phi_1(X_1) & \dots & \phi_n(X_1) \\ \phi_0(X_2) & \phi_1(X_2) & \dots & \phi_n(X_2) \\ \phi_0(X_3) & \phi_1(X_3) & \dots & \phi_n(X_3) \\ \phi_0(X_4) & \phi_1(X_4) & \dots & \phi_n(X_4) \\ \phi_0(X_5) & \phi_1(X_5) & \dots & \phi_n(X_5) \\ \phi_0(X_6) & \phi_1(X_6) & \dots & \phi_n(X_6) \\ \phi_0(X_7) & \phi_1(X_7) & \dots & \phi_n(X_7) \\ \vdots & \vdots & & \vdots \\ \phi_0(X_m) & \phi_1(X_m) & \dots & \phi_n(X_m) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \approx \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \\ U_7 \\ \vdots \\ U_m \end{bmatrix}$$

Minimisons les résidus

$$R_i = U_i - \underbrace{\sum_{j=0}^{n} \phi_j(X_i)\ a_j}_{u^h(X_i)} \qquad i = 0, 1, 2, \dots, m.$$



Problème de l'approximation

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

$$\underbrace{\sum_{i=0}^{m} \left(U_i - \sum_{j=0}^{n} \phi_j(X_i)\ a_j \right)^2}_{J(a_0, \ldots, a_n)} \quad \text{soit minimal.}$$

Il s'agit donc de minimiser J qui est une fonction à n+1 variables

$$\left. \frac{\partial J}{\partial a_k} \right|_{(a_0, \ldots, a_n)} = 0$$

Il faut que le vecteur gradient s'annule et aussi que les conditions du second ordre sur la matrice hessienne soient satisfaites

Calculons…

$$\underbrace{\sum_{i=0}^{m} \left(U_i - \sum_{j=0}^{n} \phi_j(X_i)\ a_j \right)^2}_{J(a_0, \ldots, a_n)}$$

$$\left. \frac{\partial J}{\partial a_k} \right|_{(a_0, \ldots, a_n)} = 0 \qquad k = 0, 1, \ldots, n$$

$$\downarrow$$

$$\sum_{i=0}^{m} -2\phi_k(X_i) \left(U_i - \sum_{j=0}^{n} \phi_j(X_i)\ a_j \right) = 0 \qquad k = 0, 1, \ldots, n$$

$$\downarrow$$

$$\sum_{i=0}^{m} \phi_k(X_i) \sum_{j=0}^{n} \phi_j(X_i)\ a_j = \sum_{i=0}^{m} \phi_k(X_i) U_i \qquad k = 0, 1, \ldots, n$$

$$\downarrow$$

$$\sum_{j=0}^{n} \left(\sum_{i=0}^{m} \phi_k(X_i)\ \phi_j(X_i) \right) a_j = \sum_{i=0}^{m} \phi_k(X_i) U_i \qquad k = 0, 1, \ldots, n$$

Et pratiquement…

$$\begin{bmatrix} \sum_{i=0}^{m}\phi_0(X_i)\phi_0(X_i) & \sum_{i=0}^{m}\phi_0(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^{m}\phi_0(X_i)\phi_n(X_i) \\ \sum_{i=0}^{m}\phi_1(X_i)\phi_0(X_i) & \sum_{i=0}^{m}\phi_1(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^{m}\phi_1(X_i)\phi_n(X_i) \\ \sum_{i=0}^{m}\phi_2(X_i)\phi_0(X_i) & \sum_{i=0}^{m}\phi_2(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^{m}\phi_2(X_i)\phi_n(X_i) \\ & \vdots & & \vdots \\ \sum_{i=0}^{m}\phi_n(X_i)\phi_0(X_i) & \sum_{i=0}^{m}\phi_n(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^{m}\phi_n(X_i)\phi_n(X_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{m}\phi_0(X_i)U_i \\ \sum_{i=0}^{m}\phi_1(X_i)U_i \\ \sum_{i=0}^{m}\phi_2(X_i)U_i \\ \vdots \\ \sum_{i=0}^{m}\phi_n(X_i)U_i \end{bmatrix}$$

Equations normales

A_{ik}

A^T_{ki}

A_{ij}

Problème de l'approximation

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

$$\sum_{j=0}^{n}\left(\sum_{i=0}^{m}\phi_k(X_i)\phi_j(X_i)\right)a_j = \sum_{i=0}^{m}\phi_k(X_i)U_i \qquad k = 0, 1, \dots, n$$

Equations normales sous forme matricielle $A^T A\, a = A^T u$

Problème de l'interpolation

A est une matrice carrée

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

Equations de l'interpolation sous forme matricielle $A\,a = u$

On voit bien que dans le cas n=m, les équations normales fournissent la même solution que les équations de l'interpolation…

Cas particulier : Régression polynomiale

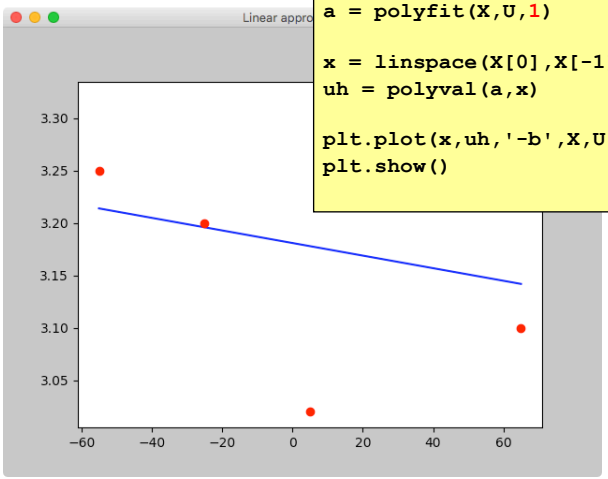
$$\phi_j(x) = x^j \qquad j = 0, 1, 2, \dots, n.$$

$$u^h(x) = a_0 + a_1 x + a_2 x^2$$

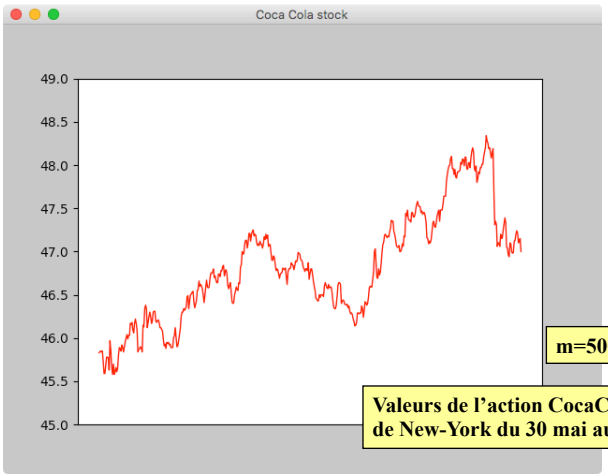
$$\begin{bmatrix} (m+1) & \sum_{i=0}^{m} X_i & \sum_{i=0}^{m} X_i^2 \\ \sum_{i=0}^{m} X_i & \sum_{i=0}^{m} X_i^2 & \sum_{i=0}^{m} X_i^3 \\ \sum_{i=0}^{m} X_i^2 & \sum_{i=0}^{m} X_i^3 & \sum_{i=0}^{m} X_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{m} U_i \\ \sum_{i=0}^{m} X_i U_i \\ \sum_{i=0}^{m} X_i^2 U_i \end{bmatrix}$$

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,1)

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

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



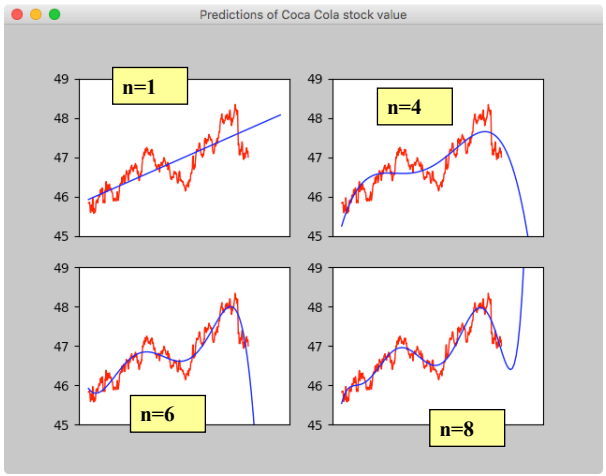
Beaucoup de données



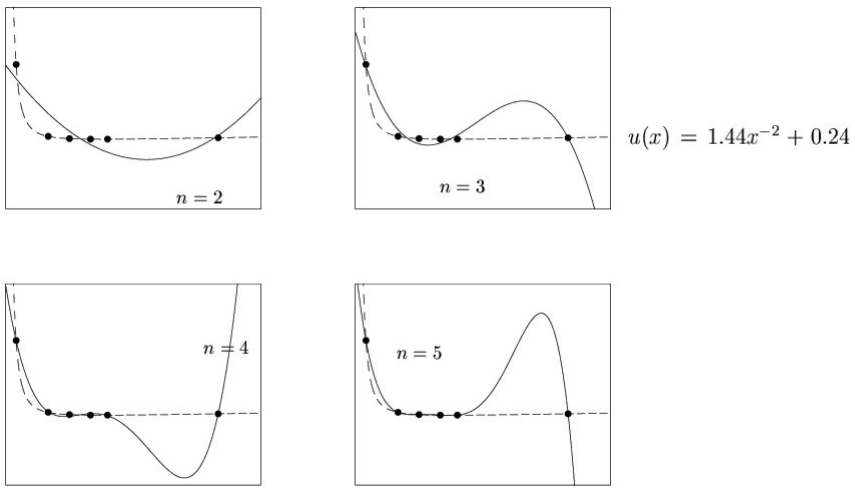
m=500

Valeurs de l'action CocaCola à la bourse de New-York du 30 mai au 18 juin 2003

Faisons des régressions et extrapolons les valeurs pour devenir riches :-)



…les approximations polynomiales divergent aussi !



$u(x) = 1.44x^{-2} + 0.24$

Et si nous avions la fonction u(x)…

$$J(a_0, a_1, \ldots, a_n) = \int_a^b \left(u(x) - \sum_{j=0}^{n} \phi_j(x)\ a_j \right)^2 dx$$

$$\left.\frac{\partial J}{\partial a_k}\right|_{(a_0,\ldots,a_n)} = 0 \qquad k = 0, 1, \ldots, n$$

$$\int_a^b -2\phi_k(x) \left(u(x) - \sum_{j=0}^{n} \phi_j(x)\ a_j \right) dx = 0 \qquad k = 0, 1, \ldots, n$$

$$\sum_{j=0}^{n} \left(\int_a^b \phi_k(x)\phi_j(x)dx \right) a_j = \int_a^b \phi_k(x)u(x)dx \qquad k = 0, 1, \ldots, n$$

$$B\ a = c$$

Problème intégral de l'approximation

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

$$\sum_{j=0}^{n} \left(\int_a^b \phi_k(x)\phi_j(x)dx \right) a_j = \int_a^b \phi_k(x)u(x)dx \qquad k = 0, 1, \ldots, n$$

Les équations normales ne sont qu'une approximation de la forme intégrale par une somme finie

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

$$\sum_{j=0}^{n} \left(\sum_{i=0}^{m} \phi_k(X_i)\ \phi_j(X_i) \right) a_j = \sum_{i=0}^{m} \phi_k(X_i) U_i \qquad k = 0, 1, \ldots, n$$

Et pratiquement…

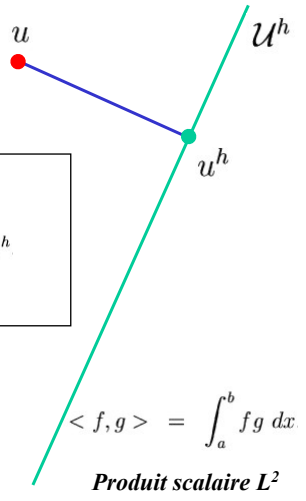
$$\begin{bmatrix} \int_a^b \phi_0(x)\phi_0(x)dx & \int_a^b \phi_0(x)\phi_1(x)dx & \dots & \int_a^b \phi_0(x)\phi_n(x)dx \\ \int_a^b \phi_1(x)\phi_0(x)dx & \int_a^b \phi_1(x)\phi_1(x)dx & \dots & \int_a^b \phi_1(x)\phi_n(x)dx \\ \int_a^b \phi_2(x)\phi_0(x)dx & \int_a^b \phi_2(x)\phi_1(x)dx & \dots & \int_a^b \phi_2(x)\phi_n(x)dx \\ & \vdots & & \vdots \\ \int_a^b \phi_n(x)\phi_0(x)dx & \int_a^b \phi_n(x)\phi_1(x)dx & \dots & \int_a^b \phi_n(x)\phi_n(x)dx \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \int_a^b \phi_0(x)u(x)dx \\ \int_a^b \phi_1(x)u(x)dx \\ \int_a^b \phi_2(x)u(x)dx \\ \vdots \\ \int_a^b \phi_n(x)u(x)dx \end{bmatrix}$$

Calcul de la projection orthogonale de u(x) pour le produit scalaire L^2

L'approximation au sens des moindres carrés est une projection orthogonale…

$$< \widehat{u}^h , \underbrace{(u - u^h)}_{e^h} > \; = \; 0, \qquad \forall \widehat{u}^h \in \mathcal{U}^h$$

Sous-espace vectoriel dont une base est donnée par les fonctions ϕ_i



$$< f, g > \; = \; \int_a^b f\, g \; dx$$

Produit scalaire L^2

Et il s'agit bien du même $u^h(x)$.

$$< \widehat{u}^h \, , \underbrace{(u - u^h)}_{e^h} > \;=\; 0, \qquad \forall \widehat{u}^h \in \mathcal{U}^h,$$

Imposer pour toute fonction de base ϕ_i est équivalent à l'imposer pour toute fonction de l'espace

$$\int_a^b \phi_k(x) \left(u(x) - \sum_{j=0}^{n} \phi_j(x) \; a_j \right) dx \;=\; 0, \qquad k = 0, 1, \ldots, n$$

$$\sum_{j=0}^{n} \left(\int_a^b \phi_k(x) \phi_j(x) dx \right) \; a_j \;=\; \int_a^b \phi_k(x) u(x) dx \qquad k = 0, 1, \ldots, n$$

Equations intégrales de l'approximation au sens des moindres carrés !

□

Une base orthogonale pour ce produit scalaire !

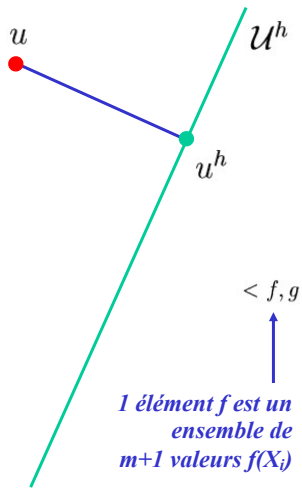
$$\phi_i(x) \;=\; \cos(ix) \qquad i = 1, 2, \ldots, n$$

$[-\pi, \pi]$.

$$\int_{-\pi}^{\pi} \cos(ix) \cos(jx) \;=\; 0 \qquad i \neq j$$
$$\;=\; \pi \qquad i = j$$

$$a_i = \frac{1}{\pi} \int_{-\pi}^{\pi} u(x) \cos(ix) dx$$

Sous-espace vectoriel dont une base est donnée par les n+1 vecteurs $\phi_j(X_i)$



Approximer m+1 points au sens des moindres carrés avec n+1 fonctions de base

$$< f, g >_* \;=\; \sum_{j=0}^{m} f(X_j) g(X_j)$$

1 élément f est un ensemble de m+1 valeurs $f(X_i)$

C'est réaliser une projection orthogonale d'un élément de R^{m+1} dans un sous-espace de dimension n+1

En fait, on réalise deux approximations successives…

Cet élément de R^m représente toutes les fonctions dont les m valeurs en X_i sont identiques

$$< \widehat{u}^h \;,\; \underbrace{(u - u^h)}_{e^h} >_* \;=\; 0, \qquad \forall \widehat{u}^h \in \mathcal{U}^h,$$

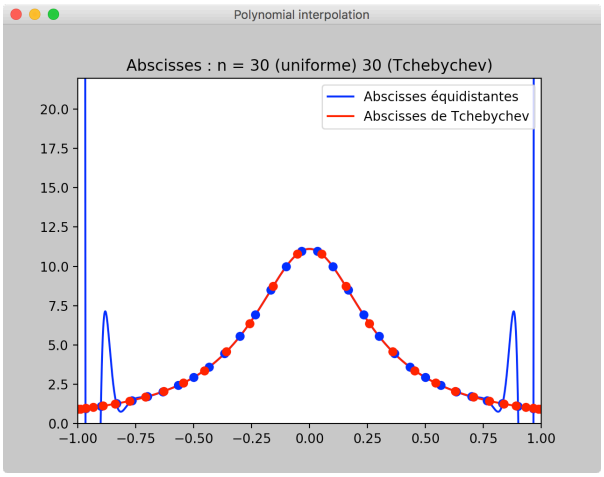
$$\sum_{i=0}^{m} \phi_k(X_i) \left(u(X_i) - \sum_{j=0}^{n} \phi_j(X_i)\; a_j \right) \;=\; 0, \qquad k = 0, 1, \ldots, n$$

$$\sum_{j=0}^{n} \left(\sum_{i=0}^{m} \phi_k(X_i)\; \phi_j(X_i) \right) a_j \;=\; \sum_{i=0}^{m} \phi_k(X_i) U_i \qquad k = 0, 1, \ldots, n$$

Equations normales !

□

Implémenter l'interpolation polynomiale…



Une implémentation naïve mais qui fournit la bonne réponse…

```
def lagrange_naive(x,X,U):

  n = min(len(X),len(U))
  m = len(x)
  uh = zeros(m)
  phi = zeros(n)
  for j in range(m):
    uh[j] = 0
    for i in range(n):
      phi[i] = 1.0
      for k in range(n):
        if i != k:
          phi[i] = phi[i]*(x[j]-X[k])/(X[i]-X[k])
      uh[j] = uh[j] + U[i] * phi[i]
  return uh
```

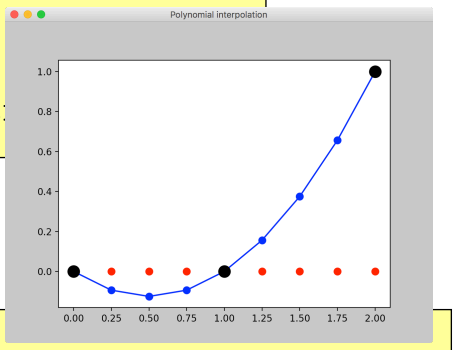
Pas si naïf que cela :
Implémentation robuste !
Pas d'erreur possible !

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

```
def lagrange_naive(x,X,U):

  n = min(len(X),len(U))
  m = len(x)
  uh = zeros(m)
  phi = zeros(n)
  for j in range(m):
    uh[j] = 0
    for i in range(n):
      phi[i] = 1.0
      for k in range(n):
        if i != k:
          phi[i] = phi[i]*(x[j]-X[k])/(X[i]-X[k])
      uh[j] = uh[j] + U[i] * phi[i]
  return uh
```

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

```
def lagrange_naive(x,X,U):

  n = min(len(X),len(U))
  ...
  for j in range(m):
    uh[j] = 0
    for i in range(n):
      ...
      uh[j] = uh[j] + U[i] * phi[i]
  return uh
```



x : c'est quoi ?

```
X = [0,1,2]
U = [0,0,1]
x = linspace(0,2,9)
uh = lagrange_naive(x,X,U);

plt.plot(x,zeros(size(x)),'.r',markersize=15)
plt.plot(x,uh,'.-b',markersize=15)
plt.plot(X,U,'.k',markersize=25)
```

Pourquoi est-ce naïf ?

```
x = linspace(-1,1,2000)
tic()
lagrange(x,[0,1,2],[0,1,2])
toc()
tic()
lagrange(x,[0,1,2],[0,1,2
toc()
```

```
Elapsed time is 0.000219 seconds
Elapsed time is 0.017949 seconds
```

```
x = linspace(-1,1,2000000
tic()
lagrange_naive(x,[0,1,2],[0,1,2])
...
```

```
Elapsed time is 1.415670 seconds
Elapsed time is 159.760674 seconds
```

CTRL-C

Pas de vectorisation :-(
Mais pré-allocation :-)

Suite de Pisano Fibonacci

A man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair if it is supposed that every month each pair begets a new pair which from the second month on becomes productive…

$$f_n = f_{n-1} + f_{n-2}$$

```
def fibonacci(n):
  f = zeros(n)
  f[0] = 1
  f[1] = 2
  for k in range(2,n):
    f[k] = f[k-1] + f[k-2]
  return f
```

Dans numpy, il faut toujours préallouer les tableaux…

```
def fibonacci(n):
  f = zeros(n)
  f[0] = 1
  f[1] = 2
  for k in range(2,n):
    f[k] = f[k-1] + f[k-2]
  return f
```

```
def fibonaccon(n):

  f[0] = 1
  f[1] = 2
  for k in range(2,n):
    f[k] = f[k-1] + f[k-2]
  return f
```

```
bash-3.2$ python fibonacci.py
Traceback (most recent call last):
  File "fibonacci.py", line 69, in <module>
    fibonacci(n)
  File "fibonacci.py", line 50, in fibonacci
    f[0] = 1
NameError: name 'f' is not defined
```

Et si tu veux vraiment pas pré-allouer les tableaux…

```
def fibonacci(n):
  f = zeros(n)
  f[0] = 1
  f[1] = 2
  for k in range(2,n):
    f[k] = f[k-1] + f[k-2]
  return f
```

```
def fibonaccon(n):

  f = [1,2]

  for k in range(2,n):
    f = append(f,[f[k-1] + f[k-2]])
  return f
```

```
bash-3.2$ python fibonacci.py
Elapsed time is 0.070130 seconds  (fibonacci)
Elapsed time is 10.754738 seconds  (fibonaccon)
```

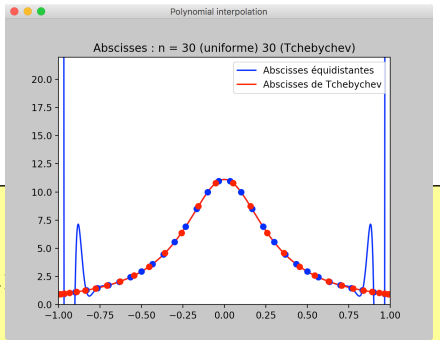
```
n = 200000
tic()
fibonacci(n)
toc('fibonacci')
```

Ecrire le maximum d'opérations sous forme matricielle

```
U @ phi
```

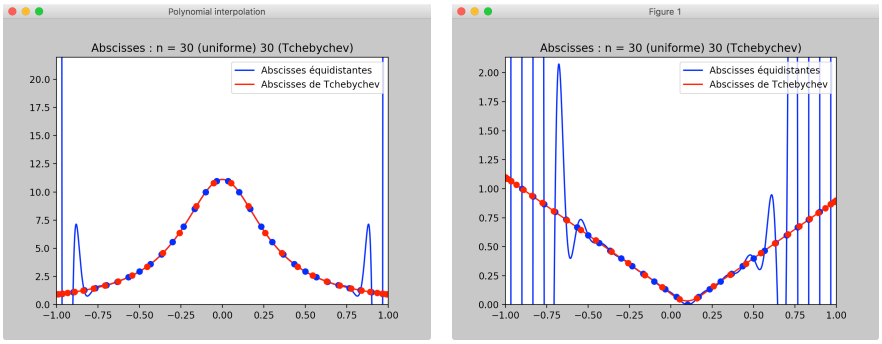
```
def lagrange(x,X,U):

  n = min(len(X),len(U))        # Evite un message d'erreur :-)
  phi = ones((n,len(x)))        # Preallocation (imposé !)
  for i in range(n):
    for k in list(range(0,i)) + list(range(i+1,n)):
      phi[i,:] = phi[i,:]*(x-X[k])/(X[i]-X[k])

  return U @ phi                # Produit matriciel (plus rapide !)
```

Obtenir la figure !




```
h = 1/n
x = linspace(-1,1,2000)
Xunif = arange(-1+h/2,1+h/2,h)
Xcheb = cos( pi*(2*arange(0,2*n)+1

u = lambda x : 1/(x**2 + 0.09)
Uunif = u(Xunif)
Ucheb = u(Xcheb)

plt.figure('Polynomial interpolation')
plt.plot(x,lagrange(x,Xunif,Uunif),'-b',label='Absc. équidistantes')
plt.plot(x,lagrange(x,Xcheb,Ucheb),'-r',label='Absc. de Tchebychev')
plt.plot(Xunif,Uunif,'ob',Xcheb,Ucheb,'or')
plt.xlim((-1,1))
plt.ylim((0,max(Uunif)*2))
plt.title('Abscisses : n = %d (uniforme) %d (Tchebychev)'
            % (len(Xunif),len(Xcheb)))
plt.legend(loc='upper right')
```

Obtenir deux figures…



Cool : ctrl-c / ctrl-v ☺

Ne jamais dupliquer du code !

```
functions = [lambda x : 1/(x**2 + 0.09),
             lambda x : abs(x - 0.1)]

for u in functions:
  Uunif = u(Xunif)
  Ucheb = u(Xcheb)
  plt.figure('Polynomial interpolation')
  plt.plot(x,lagrange(x,Xunif,Uunif),'-b', ...
  plt.plot(x,lagrange(x,Xcheb,Ucheb),'-r', ...
```

Les listes de Python peuvent contenir n'importe quoi et donc aussi des fonctions

Il faudra maintenir deux fois plus de lignes de code : c'est cher !

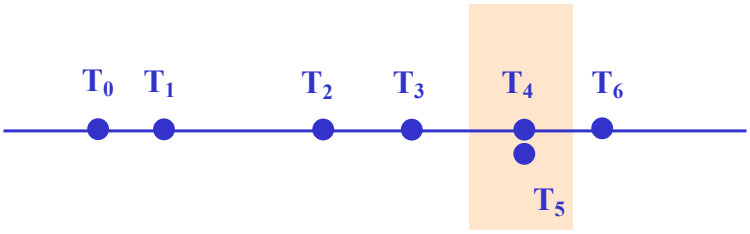
Introduction aux NURBS

The Art of 3D Computer Modeling

www.pixar.com

Des nœuds…

$$T_0 \leq T_1 \leq T_2 \leq \ldots \leq T_n,$$



Un nœud est de multiplicité m s'il est répété m fois

Fonctions B-splines

Définition 1.4.

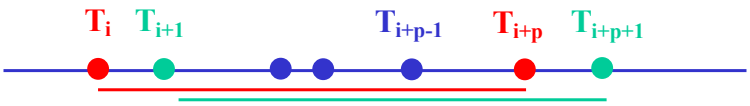
Soit les $n+1$ noeuds $T_0 \leq T_1 \leq T_2 \leq \ldots \leq T_n$.
Les B-splines de degré p sont les $n-p$ fonctions $B_i^p(t)$. Une fonction B_i^p est nulle sauf dans l'intervalle $[\, T_i, T_{i+1+p}\, [$ où elle est définie par la relation de récurrence :

$$B_i^p(t) = \frac{(t-T_i)}{(T_{i+p}-T_i)} B_i^{p-1}(t) + \frac{(T_{i+1+p}-t)}{(T_{i+1+p}-T_{i+1})} B_{i+1}^{p-1}(t)$$

avec $i = 0, \ldots, n-p-1$ et en partant de :

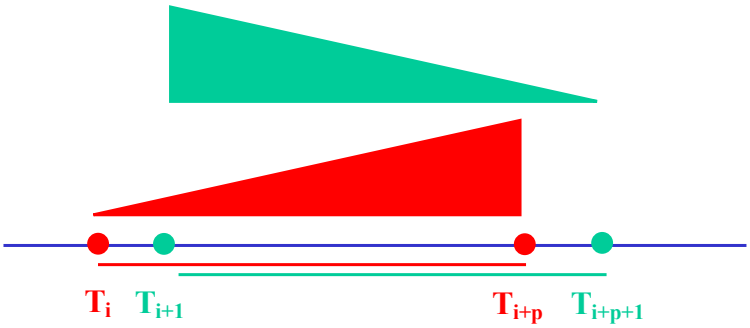
$$B_i^0(t) = \begin{cases} 1, & si \;\; t \in [T_i, T_{i+1}[\\ 0, & ailleurs \end{cases}$$

On observe immédiatement que pour des noeuds de multiplicité supérieure à un, la formule de récurrence peut faire apparaître une valeur nulle à l'un ou l'autre dénominateur. On complète donc la définition en spécifiant qu'il ne faut tenir compte que des termes dont les dénominateurs ne s'annulent pas.

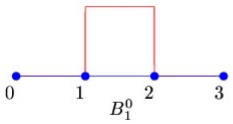


Les B-splines ou fonctions de mélange

$$B_i^p(t) = \frac{(t-T_i)}{(T_{i+p}-T_i)} B_i^{p-1}(t) + \frac{(T_{i+1+p}-t)}{(T_{i+1+p}-T_{i+1})} B_{i+1}^{p-1}(t)$$



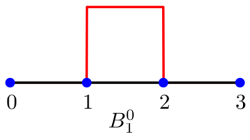
Commençons la récurrence…

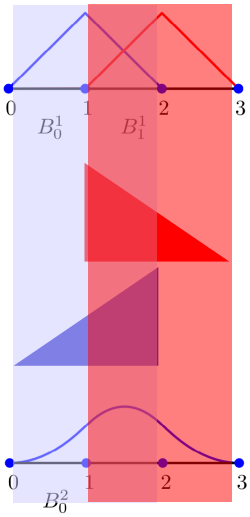


$$B_i^0(t) = \begin{cases} 1, & si \ \ t \in [T_i T_{i+1}[\\ 0, & ailleurs \end{cases}$$

$$B_i^p(t) = \frac{(t - T_i)}{(T_{i+p} - T_i)} B_i^{p-1}(t) + \frac{(T_{i+1+p} - t)}{(T_{i+1+p} - T_{i+1})} B_{i+1}^{p-1}(t)$$

$$B_i^p(t) = \boxed{\frac{(t - T_i)}{(T_{i+p} - T_i)} B_i^{p-1}(t)} + \boxed{\frac{(T_{i+1+p} - t)}{(T_{i+1+p} - T_{i+1})} B_{i+1}^{p-1}(t)}$$

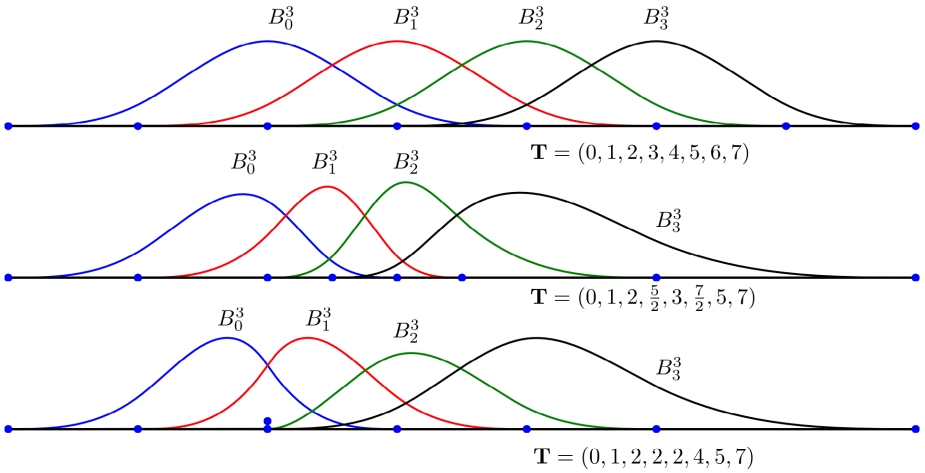


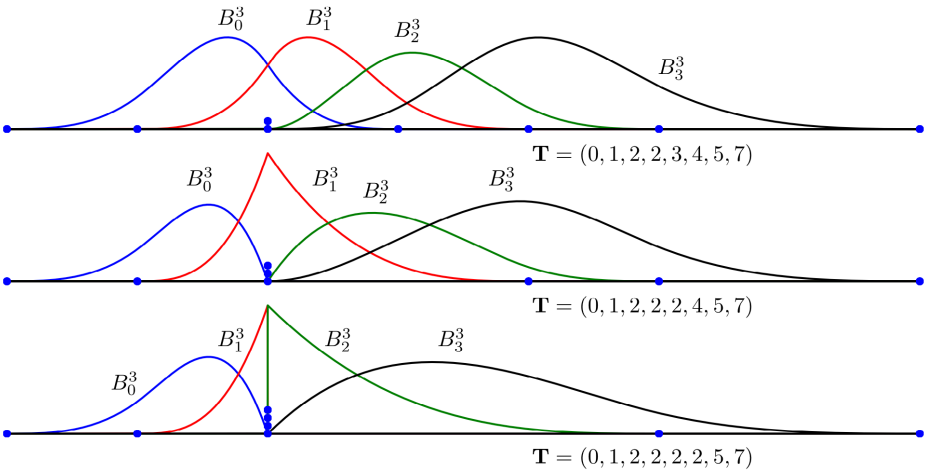


Exemple facile…

On peut observer que la fonction B-spline est toujours comprise entre 0 et 1

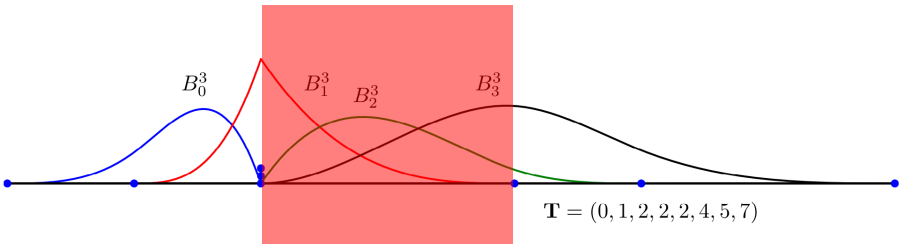
Modifions les noeuds…





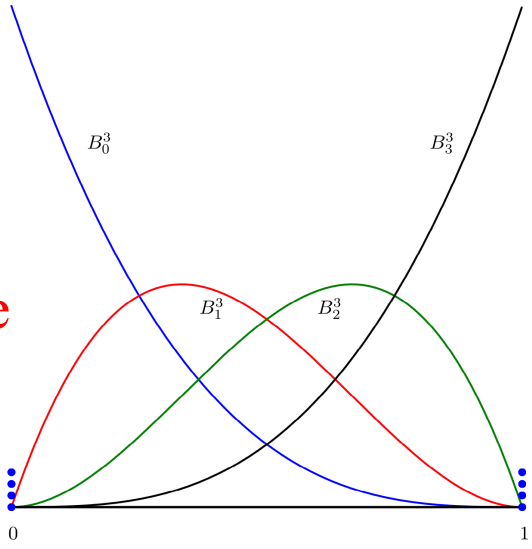
Nœuds doubles, triples…

Observons



- Dans l'intervalle $[\ T_i, T_{i+1}\ [$, seules les splines $B_{i-p}^p(t), \ldots, B_i^p(t)$ sont non nulles.
- Une spline $B_i^p(t)$ ne vaut exactement 1 qu'en un noeud de multiplicité supérieure ou égale à p.

Quand les NURBS deviennent des splines de Bézier qui sont des polynômes de Bernstein



Pierre Bézier

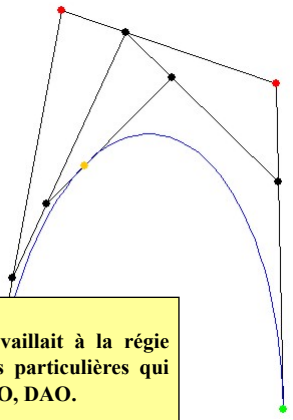
Né le 1er septembre 1910 à Paris
Décédé le 25 janvier 1999

In 1933, Bezier entered Renault and worked for this company for 42 years. In 1948, as Director of Production Engineering he was responsible for the design of the transfer lines producing most of the 4 CV mechanical parts. In 1957, he became Director of Machine Tool Division and was responsible for the automatic assembly of mechanical components, and for the design and production of an NC drilling and milling machine.

Bezier started his research in CADCAM in 1960 when he devoted a substantial amount of his time working on his UNISURF system.
His system was launched in 1968.

A la mémoire de
mon grand-père, ingénieur mécanicien,
mon père, ingénieur mécanicien,
ma sœur, ingénieur chimiste, docteur ès sciences,
P.B.

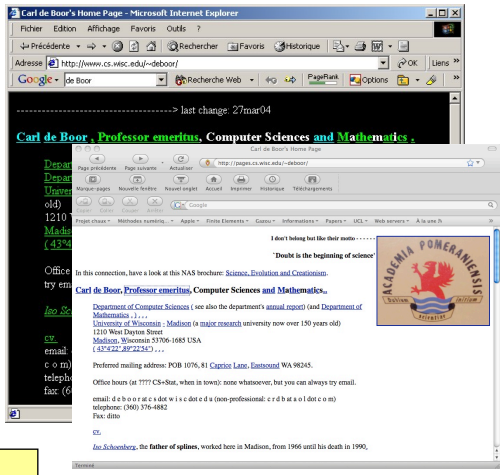
Paul Faget de Casteljau



Pierre Bezier, ingénieur des Arts et Métiers, travaillait à la régie Renault, quand il "inventa" en 1970 des courbes particulières qui portent maintenant son nom. Ce fut le début de CAO, DAO.

Au départ, il devait trouver un moyen simple pour reproduire sur ordinateur des courbes tracées à main-levée par des dessinateurs. Il retravailla certaines études de Paul Faget de Casteljau, ingénieur travaillant chez Citroën, c'est pourquoi on utilise "l'algorithme de Casteljau" pour les tracer. Aujourd'hui ses courbes sont utilisées dans le monde entier sans vraiment savoir d'où vient cette appellation.

Carl de Boor



Généralisation de l'algorithme de Casteljau pour des NURBS :
Algorithme de Carl de Boor

http://www.cs.wisc.edu/~deboor/

Les B-splines forment une partition unité

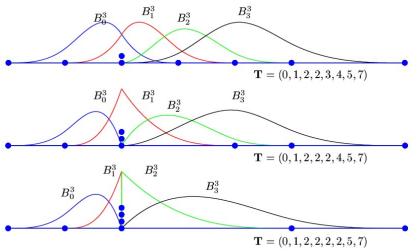
A l'exclusion des p premiers et p derniers intervalles, la somme des B-splines vaut l'unité.

$$\sum_{i=0}^{n-p-1} B_i^p(t) = 1 \qquad T_p \leq t \leq T_{n-p}$$

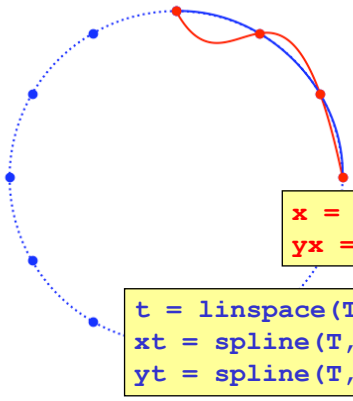
- Une spline $B_i^p(t)$ est toujours comprise entre 0 et 1.
- Une spline $B_i^p(t)$ est non nulle que dans l'intervalle $[\ T_i, T_{i+1+p}\ [$.
- Dans l'intervalle $[\ T_i, T_{i+1}\ [$, seules les splines $B_{i-p}^p(t), \ldots, B_i^p(t)$ sont non nulles.
- Une spline $B_i^p(t)$ ne vaut exactement 1 qu'en un noeud de multiplicité supérieure ou égale à p.

Continuité des B-splines

Théorème 1.4. *Si le vecteur des noeuds est constitué uniquement de points de multiplicité un, les splines de base B_i^p sont $(p-1)$-fois dérivables. On dit que l'ordre de continuité est $p-1$. Par contre, à un point de multiplicité m, l'ordre de continuité n'y sera localement que de $p-m$.*



Courbes splines paramétrées



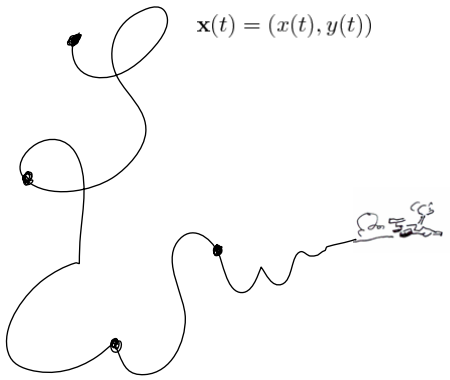
```
T = pi * arange(0,4) / 6
X = sin(T)
Y = cos(T)
```

```
x = linspace(X[0],X[-1],100)
yx = spline(X,Y)(x)
```

```
t = linspace(T[0],T[-1],100)
xt = spline(T,X)(t)
yt = spline(T,Y)(t)
```

Courbes paramétriques composées de « splines »

$$\mathbf{x}(t) = (x(t), y(t))$$



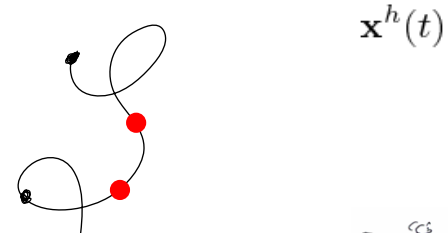
$$(X_i, Y_i) = (x(T_i), y(T_i))$$

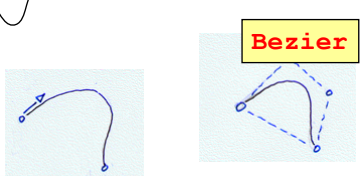
Une spline est une forme à pôles

Fonctions de base

$$\mathbf{x}^h(t) = \sum_i \phi_i(t)\,\mathbf{P}_i$$

Pôles :
- Points de passage
- Directions tangentes
- Vitesses
- Points de contrôle





Courbes composées de B-splines… ce sont des approximations !

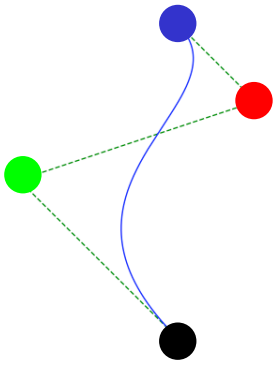
$$\begin{cases} x^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t) X_i \\ y^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t) Y_i \end{cases} \qquad T_p \leq t \leq T_{n-p}$$

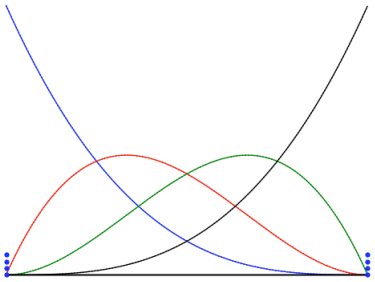
Coordonnées des points de contrôle

La courbe s'approche des points de contrôle, mais ne passe, en général, pas par ceux-ci !
Il s'agit d'une approximation, pas d'une interpolation.

Par contre, il n'y a aucun système d'équations à résoudre !

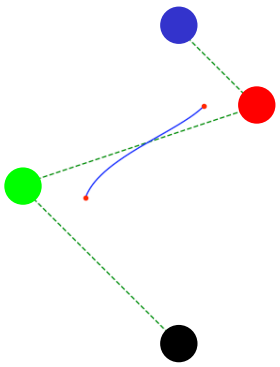
Courbe de Bezier

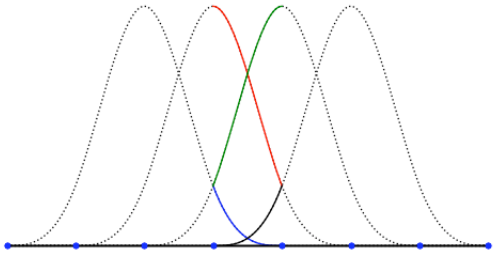




Courbe B-spline

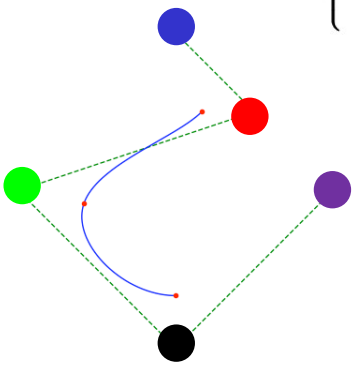
$$\begin{cases} x^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t)\ X_i \\ y^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t)\ Y_i \end{cases} \qquad T_p \leq t \leq T_{n-p}$$

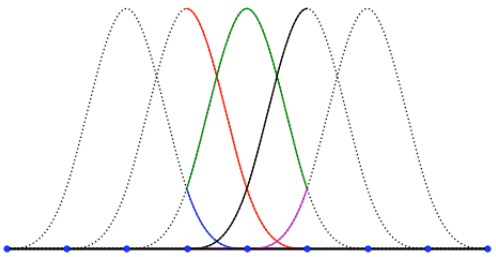




Courbe B-spline

$$\begin{cases} x^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t)\ X_i \\ y^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t)\ Y_i \end{cases} \qquad T_p \leq t \leq T_{n-p}$$

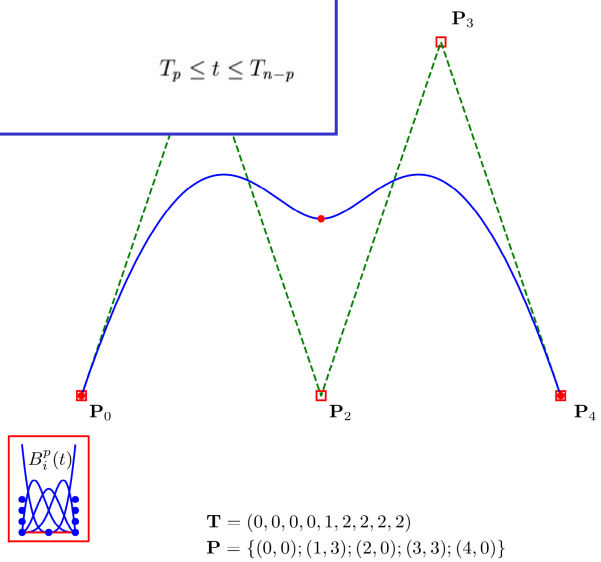




En posant $\mathbf{u}^h(t) = (x^h(t), y^h(t))$ et $\mathbf{P}_i = (X_i, Y_i)$,

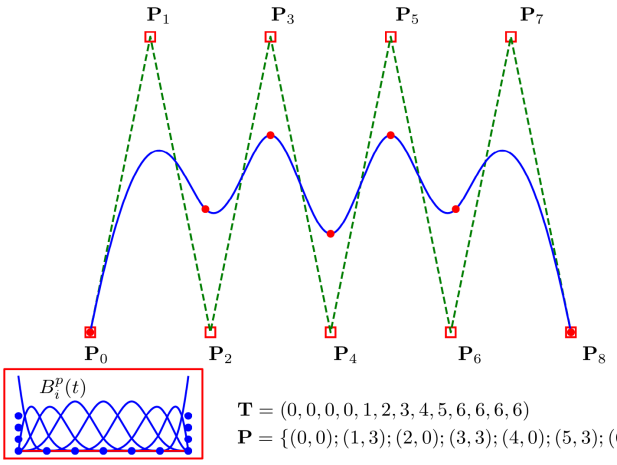
$$\mathbf{u}^h(t) = \sum_{i=0}^{n-p-1} B_i^p(t)\,\mathbf{P}_i \qquad T_p \le t \le T_{n-p}$$

B-splines



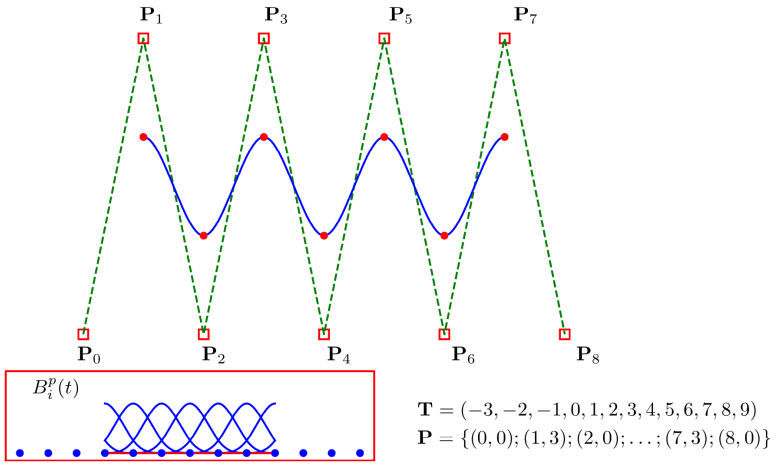
$\mathbf{T} = (0,0,0,0,1,2,2,2,2)$
$\mathbf{P} = \{(0,0);(1,3);(2,0);(3,3);(4,0)\}$

En général, on met des nœuds multiples aux extrémités…



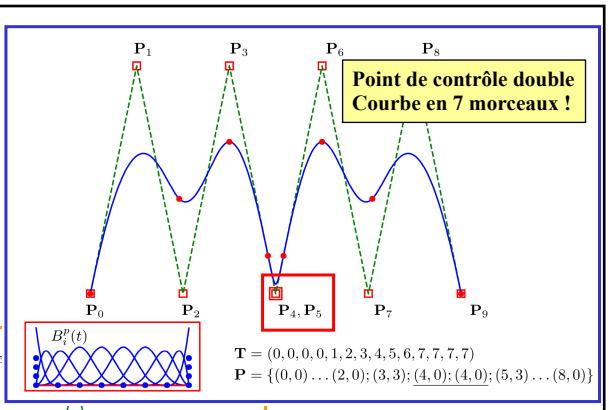
$\mathbf{T} = (0,0,0,0,1,2,3,4,5,6,6,6,6)$
$\mathbf{P} = \{(0,0);(1,3);(2,0);(3,3);(4,0);(5,3);(6,0);(7,3);(8,0)\}$

… mais ce n'est pas obligatoire !

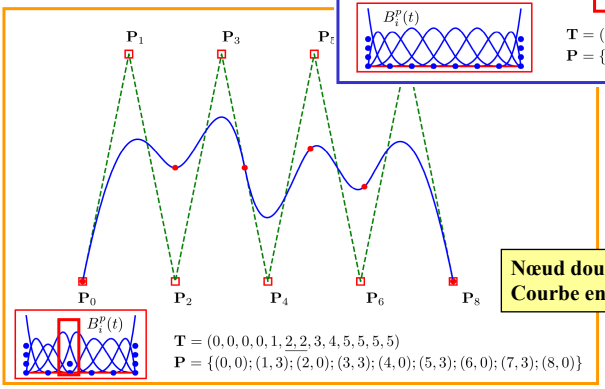


$\mathbf{T} = (-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$
$\mathbf{P} = \{(0,0); (1,3); (2,0); \ldots; (7,3); (8,0)\}$

Nœuds ou points doubles !



$\mathbf{T} = (0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 7, 7, 7)$
$\mathbf{P} = \{(0,0) \ldots (2,0); (3,3); \underline{(4,0); (4,0)}; (5,3) \ldots (8,0)\}$

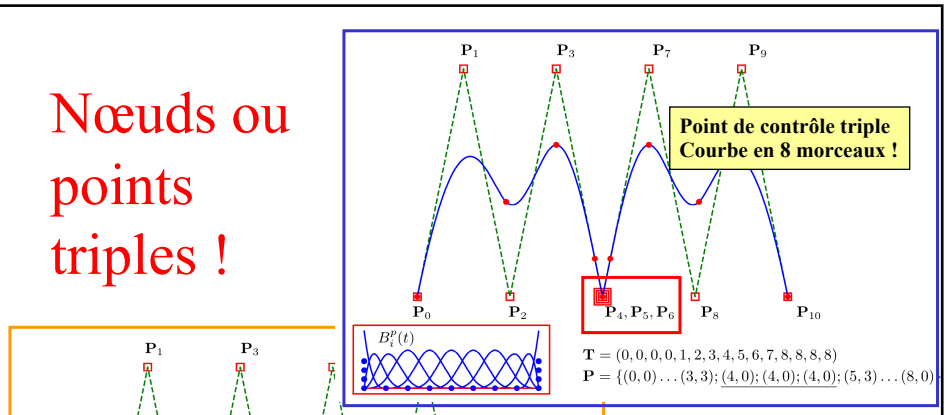


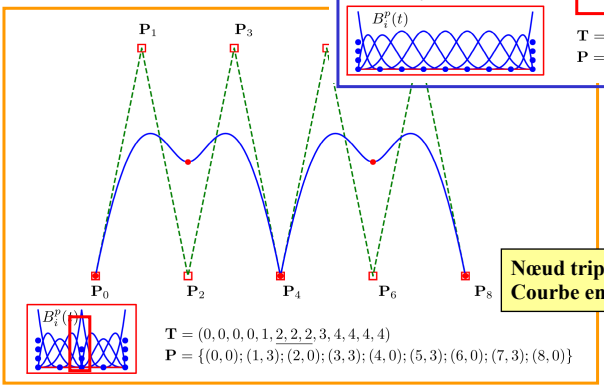
Nœuds ou points multiples ?
Effets semblables mais <u>pas</u> identiques !

Nœud double
Courbe en 5 morceaux !

$\mathbf{T} = (0, 0, 0, 0, 0, 1, \underline{2, 2}, 3, 4, 5, 5, 5, 5)$
$\mathbf{P} = \{(0,0); (1,3); (2,0); (3,3); (4,0); (5,3); (6,0); (7,3); (8,0)\}$

Nœuds ou points triples !



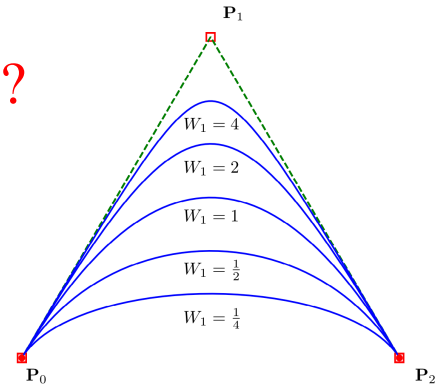


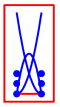
Travailler avec les nœuds est moins intuitif, mais plus efficace…

Et les NURBS ?

$$\mathbf{u}^h(t) = \frac{\sum_{i=0}^{n-p-1} W_i B_i^p(t)\ \mathbf{P}_i}{\sum_{k=0}^{n-p-1} W_k B_k^p(t)}$$

$$T_p \leq t \leq T_{n-p}$$





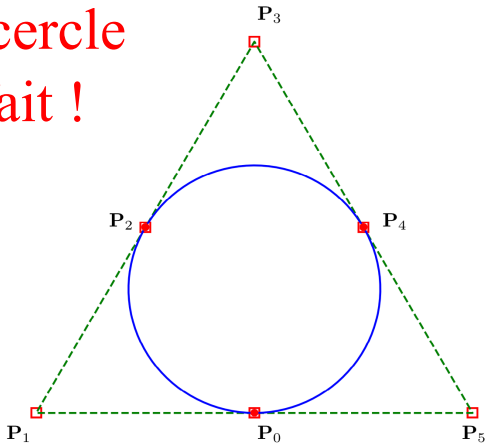
$\mathbf{T} = (0,0,0,1,1,1)$
$\mathbf{W} = (1, W_1, 1)$
$\mathbf{P} = \{(0,0);(1,\sqrt{3});(2,0)\}$

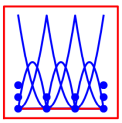
NURBS

Non-Uniform Rational B-Splines

Permet de représenter exactement toutes les coniques (ellipse, parabole, hyperbole)

Un cercle parfait !



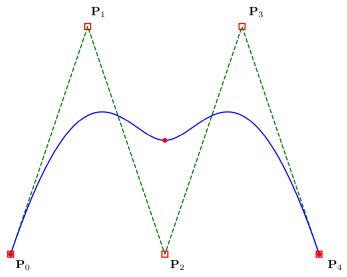


$B_i^p(t)$

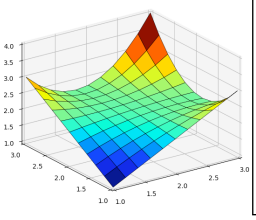
$\mathbf{T} = (0,0,0,1,1,2,2,3,3,3)$
$\mathbf{W} = (1, \frac{1}{2}, 1, \frac{1}{2}, 1, \frac{1}{2}, 1)$
$\mathbf{P} = \{(1,0);(0,0);(\frac{1}{2},\frac{\sqrt{3}}{2});(1,\sqrt{3});(\frac{3}{2},\frac{\sqrt{3}}{2});(2,0);(1,0)\}$

$$\mathbf{u}^h(t) = \frac{\sum_{i=0}^{n-p-1} W_i B_i^p(t)\ \mathbf{P}_i}{\sum_{k=0}^{n-p-1} W_k B_k^p(t)}$$

$$T_p \leq t \leq T_{n-p}$$



Généralisation à des surfaces



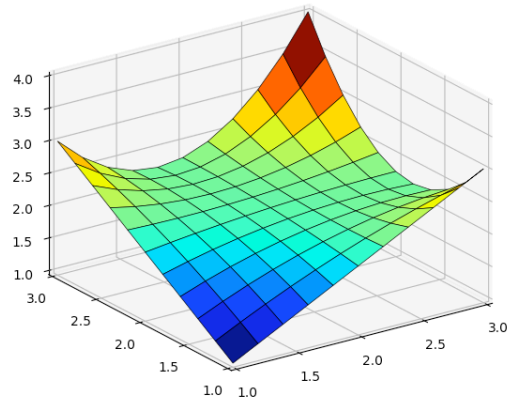
$$\mathbf{u}^h(t,s) = \frac{\sum_{i=0}^{n-p-1}\sum_{j=0}^{m-q-1} W_{ij} B_i^p(t) B_j^q(s)\ \mathbf{P}_{ij}}{\sum_{k=0}^{n-p-1}\sum_{l=0}^{m-q-1} W_{kl} B_k^p(t) B_l^q(s)}$$

$$T_p \leq t \leq T_{n-p}$$
$$S_q \leq s \leq S_{m-q}$$

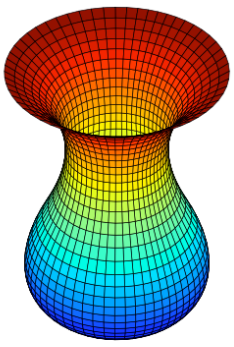
Exemple

$$\mathbf{T} = \mathbf{S} = (0,0,0,1,1,1)$$

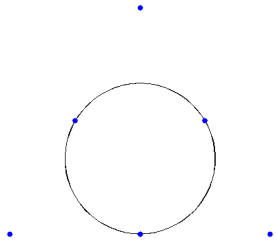
$$\mathbf{P} = \begin{bmatrix} (1,1,1) & (2,1,2) & (3,1,3) \\ (1,2,2) & (2,2,3) & (3,2,1) \\ (1,3,3) & (2,3,1) & (3,3,4) \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



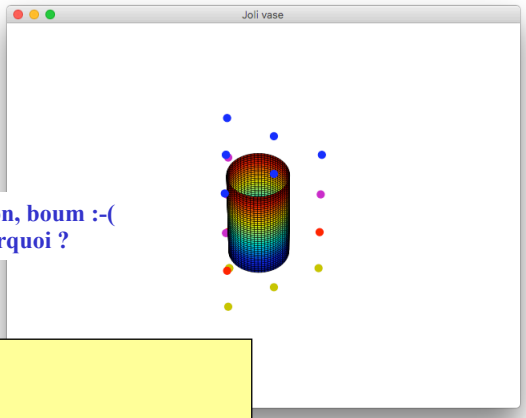
Un exemple



Potiche avec seulement 24 points de contrôle



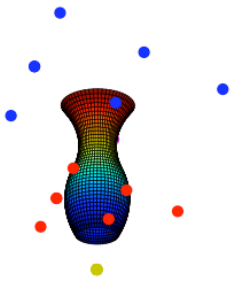
Commençons par un cylindre…

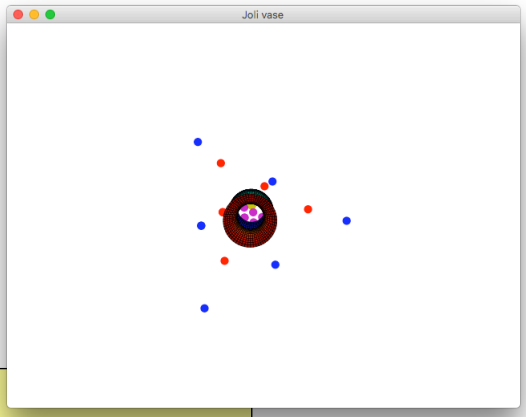


Sinon, boum :-(
Pourquoi ?

```
T = [-2,-1,0,1,2,3,4]
S = [0,0,0,1,1,2,2,3,3,4]
R = [5,5,5,5]
H = [0,5,10,15]
```

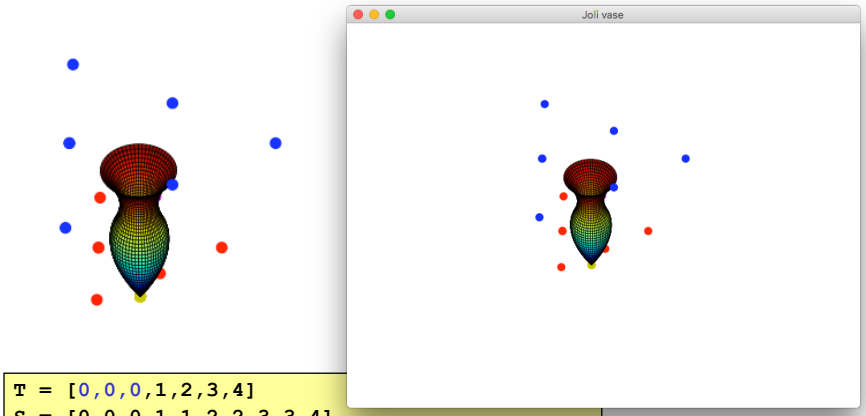
Modifions les rayons…





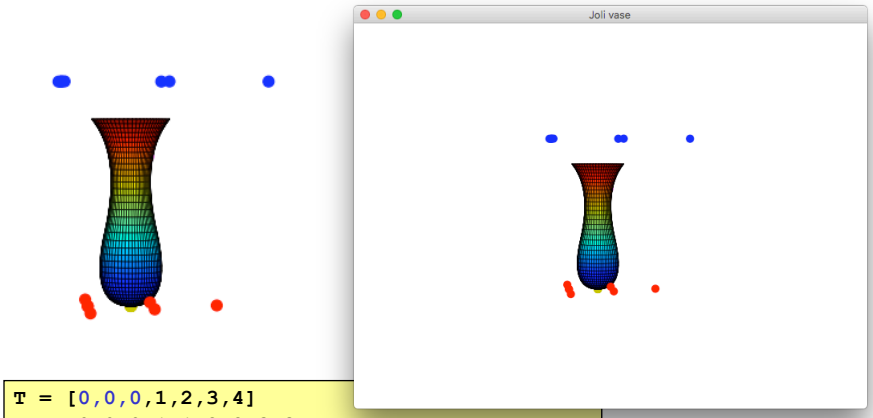

```
T = [-2,-1,0,1,2,3,4]
S = [0,0,0,1,1,2,2,3,3,4]
R = [0,5,1,8]
H = [0,5,10,15]
```

Refermons le fond du vase…




```
T = [0,0,0,1,2,3,4]
S = [0,0,0,1,1,2,2,3,3,4]
R = [0,5,1,8]
H = [0,5,10,15]
```

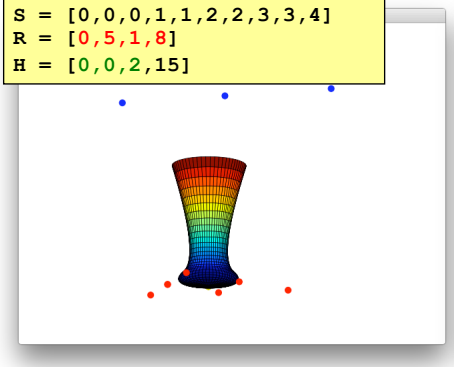
Rendons le fond plat…

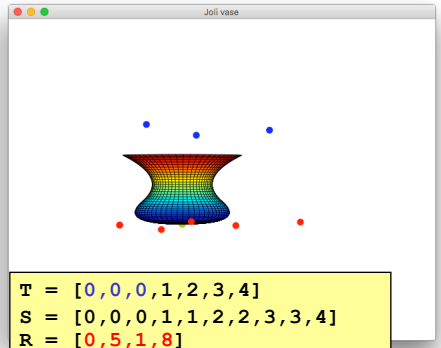



```
T = [0,0,0,1,2,3,4]
S = [0,0,0,1,1,2,2,3,3,4]
R = [0,5,1,8]
H = [0,0,10,15]
```

Et d'autres potjes ?

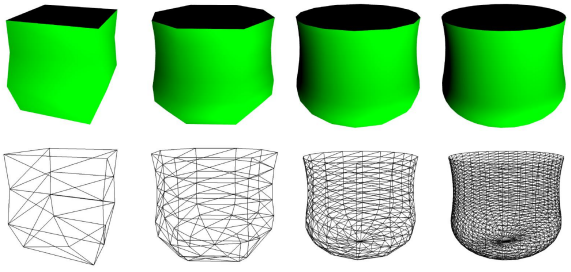
```
T = [0,0,0,1,2,3,4]
S = [0,0,0,1,1,2,2,3,3,4]
R = [0,5,1,8]
H = [0,0,2,15]
```






```
T = [0,0,0,1,2,3,4]
S = [0,0,0,1,1,2,2,3,3,4]
R = [0,5,1,8]
H = [0,0,2,4]
```

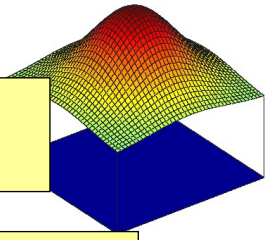
Représentation adéquate de potiches par des facettes planes…



2 triangles par patch

128 triangles par patch

Plan du cours de méthodes numériques



Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non linéaires ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Intégration numérique

Quadrature :
On estime l'intégrale définie I en effectuant une somme pondérée des valeurs $u(X_i)$

$$I = \int_a^b u(x)\,dx$$

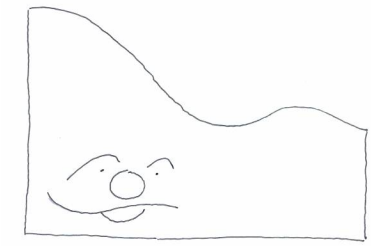
Abscisses d'intégration calculées a priori

$$I \approx I^h = \sum_{i=0}^{m} w_i\, u(X_i)$$

$$E^h = I - I^h$$

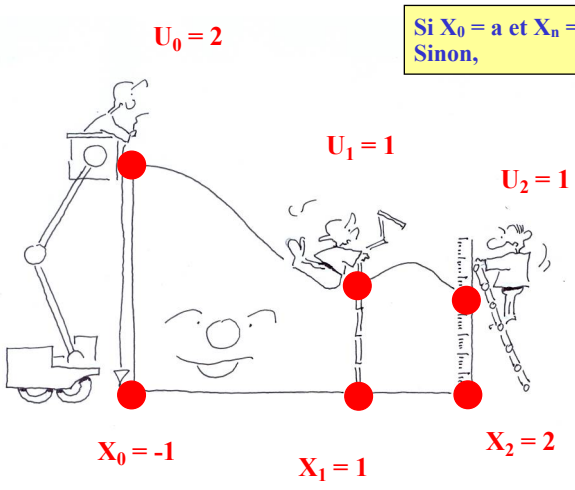
Poids calculés a priori

Une surface à intégrer….



Comment l'intégrer sur un ordinateur ?

Prise de mesures…



Si $X_0 = a$ et $X_n = b$, méthode fermée
Sinon, méthode ouverte

Simplifions, standardisons, …

$$x(\xi) = \frac{(b-a)}{2}\,\xi + \frac{(b+a)}{2}$$

$$I = \int_a^b u(x)\,dx = \int_{-1}^{1} u(x(\xi))\frac{dx}{d\xi}d\xi = \underbrace{\int_{-1}^{1} u(x(\xi))\,d\xi}_{\approx \sum_{i=0}^{m} w_i\,u(x(\Xi_i))} \frac{(b-a)}{2}$$

Méthodes d'intégration

Méthodes à pas égaux :
Règles de Newton-Cotes

Méthodes à pas inégaux :
Règles de Gauss-Legendre

Méthodes récursives :
Extrapolation de Richardson
Méthodes de Romberg

Méthodes adaptatives :
ou les méthodes numériques intelligentes…

Avec l'interpolation polynomiale, tout est facile…

$$I \approx \overbrace{\int_{-1}^{1} u^h(x)dx}^{I^h}$$

$$\downarrow$$

$$\approx \int_{-1}^{1} \sum_{i=0}^{m} u(X_i)\phi_i(x)dx$$

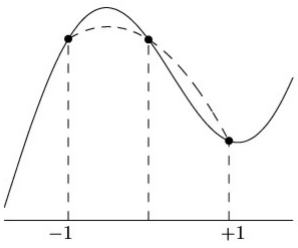
$$\downarrow$$

$$\approx \sum_{i=0}^{m} u(X_i) \underbrace{\int_{-1}^{1} \phi_i(x)dx}_{w_i}$$

$\int_{-1}^{1} u(x)dx \approx U_{-1} + U_1$	(méthode des trapèzes) $d = 1$
$\int_{-1}^{1} u(x)dx \approx \frac{1}{3} U_{-1} + \frac{4}{3} U_0 + \frac{1}{3} U_1$	(méthode de Simpson) $d = 3$
$\int_{-1}^{1} u(x)dx \approx \frac{1}{4} U_{-1} + \frac{3}{4} U_{-1/3} + \frac{3}{4} U_{1/3} + \frac{1}{4} U_1$	(méthode de Simpson $\frac{3}{8}$) $d = 3$
$\int_{-1}^{1} u(x)dx \approx \frac{7}{45} U_{-1} + \frac{32}{45} U_{-1/2} + \frac{12}{45} U_0 + \frac{32}{45} U_{1/2} + \frac{7}{45} U_1$	(méthode de Boole) $d = 5$

Quelques pages du grimoire de Gargamel

Comment obtenir les formules magiques ?



$$\int_{-1}^{1} u(x)dx \approx \overbrace{\int_{-1}^{1} u^h(x)dx}^{I^h}$$

$$\downarrow$$

$$\approx \sum_{i=0}^{2} u(X_i) \underbrace{\int_{-1}^{1} \phi_i(x)dx}_{w_i}$$

$$\downarrow$$

En fixant $X_0 = -1$, $X_1 = 0$ et $X_2 = 1$

$$\approx U(-1) \underbrace{\int_{-1}^{1} \frac{(x-1)x}{2} dx}_{w_0}$$

$$+ U(0) \underbrace{\int_{-1}^{1} \frac{(x-1)(x+1)}{-1} dx}_{w_1}$$

$$+ U(1) \underbrace{\int_{-1}^{1} \frac{(x+1)x}{2} dx}_{w_2}$$

Calcul des poids

$$w_0 = \int_{-1}^{1} \frac{(x-1)x}{2}\,dx = \left[\frac{x^3}{6} - \frac{x^2}{4}\right]_{-1}^{1} = \frac{1}{3}$$

$$w_1 = \int_{-1}^{1} 1 - x^2\,dx = \left[x - \frac{x^3}{3}\right]_{-1}^{1} = 2 - \frac{2}{3} = \frac{4}{3}$$

$$w_2 = \int_{-1}^{1} \frac{(x+1)x}{2}\,dx = \left[\frac{x^3}{6} + \frac{x^2}{4}\right]_{-1}^{1} = \frac{1}{3}$$

Avec une interpolation de degré 2, on intègre exactement un polynôme de degré 3….

$$\int_{-1}^{1} \sum_{i=0}^{2n+1} a_i\, x^i\,dx = \int_{-1}^{1} \sum_{j=0}^{n} a_{2j}\, x^{2j} dx + \underbrace{\int_{-1}^{1} \sum_{j=0}^{n} a_{2j+1}\, x^{2j+1}\,dx}_{=0}$$

$$= \int_{-1}^{1} \sum_{i=0}^{2n} a_i\, x^i\,dx$$

Une formule symétrique développée pour un degré n pair a un degré de précision n+1

L'intervalle d'intégration ne doit pas être symétrique !
Le changement de variable ne change en rien la précision de la méthode.

Introduisons le symbole h

Trapèzes $h = (b-a)$
Simpson $h = (b-a)/2$
Boole $h = (b-a)/4$

$$\int_{-h/2}^{h/2} u(x)dx \approx \frac{h}{2}\left(U_{-h/2} + U_{h/2}\right)$$
(méthode des trapèzes) $d = 1$

$$\int_{-h}^{h} u(x)dx \approx \frac{h}{3}\left(U_{-h} + 4\,U_0 + U_h\right)$$
(méthode de Simpson) $d = 3$

$$\int_{-2h}^{2h} u(x)dx \approx \frac{2h}{45}\left(7\,U_{-2h} + 32\,U_{-h} + 12\,U_0 + 32\,U_h + 7\,U_{2h}\right)$$
(méthode de Boole) $d = 5$

Méthode composite des trapèzes

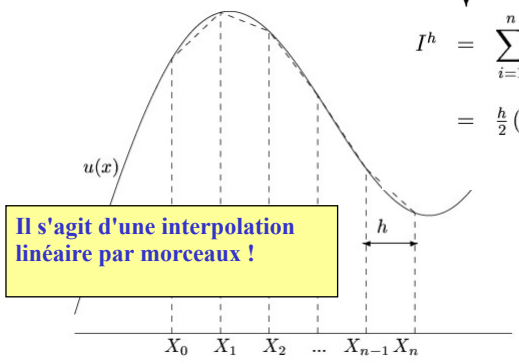
$$I = \int_{X_0}^{X_n} u(x)\,dx$$

$$= \sum_{i=1}^{n} \int_{X_{i-1}}^{X_i} u(x)\,dx$$

En utilisant la règle des trapèzes (2.7) pour chaque sous-intervalle

$$I^h = \sum_{i=1}^{n} \frac{h}{2}\left(U_{i-1} + U_i\right)$$

$$= \frac{h}{2}\left(U_0 + 2U_1 + 2U_2 + \ldots + 2U_{n-1} + U_n\right)$$



Il s'agit d'une interpolation linéaire par morceaux !

Méthode composite de Simpson

$$I = \int_{X_0}^{X_{2n}} u(x)\,dx$$

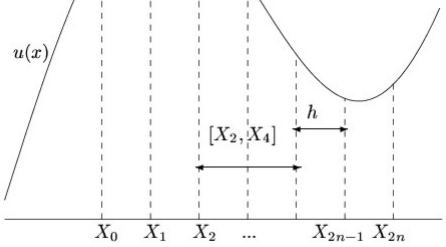
$$= \sum_{i=1}^{n} \int_{X_{2i-2}}^{X_{2i}} u(x)\,dx$$

En utilisant la règle de Simpson (2.4) pour chaque sous-intervalle

$$I^h = \sum_{i=1}^{n} \frac{h}{3}\,(U_{2i-2} + 4\,U_{2i-1} + U_{2i})$$

$$= \frac{h}{3}\,(U_0 + 4U_1 + 2U_2 + 4U_3 + 2U_4 + \ldots + 4U_{2n-1} + U_{2n})$$

Il s'agit d'une interpolation quadratique par morceaux !



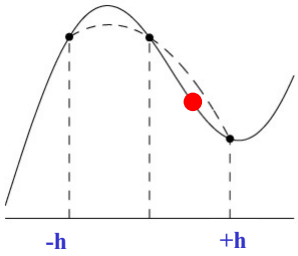
n intervalles juxtaposés de longueur 2h
2n+1 abscisses d'intégration

Comment estimer l'erreur d'intégration ?

Erreur de l'interpolation polynomiale $e^h(x)$

Erreur de l'intégration numérique $E^h(x)$

Simpson intègre parfaitement un polynôme de degré 3…



Erreur d'interpolation

$$e^h(x) = \frac{u^{(4)}(\xi)}{4!}(x-h)x(x+h)(x-\alpha)$$

où ξ est un point particulier de l'intervalle $[-h, h]$.

Ordre de précision de Simpson

$$E^h = \sum_{i=1}^{n} \int_{-h}^{h} \frac{u^{(4)}(\xi_i)}{4!}(x-h)x(x+h)(x-\alpha)\,dx$$

En définissant $C_4 = \max_i \left|u^{(4)}(\xi_i)\right|$

$$|E^h| \leq n\frac{C_4}{4!}\left|\int_{-h}^{h}(x-h)x(x+h)(x-\alpha)\,dx\right|$$

$$\leq n\frac{C_4}{4!}\left|\int_{-h}^{h}x^2(x-h)(x+h)\,dx - \alpha\underbrace{\int_{-h}^{h}(x-h)x(x+h)\,dx}_{=0}\right|$$

$$\leq n\frac{C_4}{4!}\left|\left[\frac{x^5}{5} - \frac{h^2x^3}{3}\right]_{-h}^{h}\right|$$

$$\leq n\frac{C_4}{90}h^5$$

$O(h^5)$, Monsieur ?

Nenni, nein, neen : $O(h^4)$ only !

$$\leq \quad n\frac{C_4}{90}h^5$$

Car $2nh = (b-a)$ pour une méthode composite de Simpson

$$\leq \quad (b-a)\frac{C_4}{180}h^4$$

$$|E^h| \quad \leq \quad \frac{C_4(b-a)}{180}\,h^4$$

n et h sont deux paramètres liés entre eux !
En d'autres mots, n n'est pas une constante, mais une fonction de h !

n(h) = (b-a)/2h

Degré de précision = 3
Ordre de précision = 4

Combien de sous-intervalles pour obtenir une précision donnée ?

$$\int_{-h/2}^{h/2} u(x)dx \quad = \quad \tfrac{h}{2}\,(U_{-h/2} + U_{h/2}) - \tfrac{h^3}{12}u^{(2)}(\xi)$$

$$|E^h| \quad \leq \quad \frac{C_2(b-a)}{12}\,h^2 \qquad \text{(méthode des trapèzes)} \quad \mathcal{O}(h^2)$$

$$\int_{-h}^{h} u(x)dx \quad = \quad \tfrac{h}{3}\,(U_{-h} + 4\,U_0 + U_h) - \tfrac{h^5}{90}u^{(4)}(\xi)$$

$$|E^h| \quad \leq \quad \frac{C_4(b-a)}{180}\,h^4 \qquad \text{(méthode de Simpson)} \quad \mathcal{O}(h^4)$$

$$\int_{-2h}^{2h} u(x)dx \quad = \quad \tfrac{2h}{45}\,(7\,U_{-2h} + 32\,U_{-h} + 12\,U_0 + 32\,U_h + 7\,U_{2h}) - \tfrac{8h^7}{945}u^{(6)}(\xi)$$

$$|E^h| \quad \leq \quad \frac{2C_6(b-a)}{945}\,h^6 \qquad \text{(méthode de Boole)} \quad \mathcal{O}(h^6)$$

$$\frac{C_2}{12}\,n\,\frac{(b-a)^3}{n^3} \quad \leq \quad \epsilon$$

$$\sqrt{\frac{C_2(b-a)^3}{12\epsilon}} \quad \leq \quad n$$

Méthodes d'intégration

Méthodes à pas égaux :
Règles de Newton-Cotes

Méthodes à pas inégaux :
Règles de Gauss-Legendre

Méthodes récursives :
Extrapolation de Richardson
Méthodes de Romberg

Méthodes adaptatives :
ou les méthodes numériques intelligentes…

Méthodes à pas inégaux : Gauss-Legendre

Simpson sur un intervalle
Degré de précision = 3
Nombre de points = 3

$$\int_{-1}^{1} p(x)\,dx = \sum_{k=0}^{n} w_k p(X_k)\,,$$

Choisir des abscisses équidistantes n'est pas le meilleur choix !

En développant le polynôme,

$$\sum_{i=0}^{2n+1} \int_{-1}^{1} a_i\, x^i\,dx = \sum_{k=0}^{n} w_k \sum_{i=0}^{2n+1} a_i\, X_k^i$$

Gauss-Legendre sur un intervalle
Degré de précision = 2n+1
Nombre de points = n+1

En séparant les termes pairs et impairs,

$$\sum_{j=0}^{n} \int_{-1}^{1} a_{2j} x^{2j}\,dx + \sum_{j=0}^{n} \int_{-1}^{1} a_{2j+1} x^{2j+1}\,dx = \sum_{k=0}^{n} w_k \sum_{j=0}^{n} a_{2j} X_k^{2j} + \sum_{k=0}^{n} w_k \sum_{j=0}^{n} a_{2j+1} X_k^{2j+1}$$

En effectuant les intégrales,

$$\sum_{j=0}^{n} \frac{2}{(2j+1)} a_{2j} + 0 = \sum_{k=0}^{n} w_k \sum_{j=0}^{n} a_{2j} X_k^{2j} + \sum_{k=0}^{n} w_k \sum_{j=0}^{n} a_{2j+1} X_k^{2j+1}$$

Calcul des abscisses de Gauss-Legendre

$$
\begin{aligned}
(X_0^3\ w_0 + X_1^3\ w_1) &= 0,\\
(X_0^2\ w_0 + X_1^2\ w_1) &= 2/3,\\
(X_0\ w_0 + X_1\ w_1) &= 0,\\
(w_0 + w_1) &= 2,
\end{aligned}
$$

$$
w_0 = w_1 = 1,
$$

$$
-X_0 = X_1 = \frac{1}{\sqrt{3}}.
$$

$n+1$	$X_k, -X_{n-k}$	w_k, w_{n-k}
1	0.000000000000000	2.000000000000000
2	0.577350269189626	1.000000000000000
3	0.774596669241483	0.555555555555556
	0.000000000000000	0.888888888888889
4	0.861136311594053	0.347854845137454
	0.339981043584856	0.652145154862546
5	0.906179845938664	0.236926885056189
	0.538469310105683	0.478628670499366
	0.000000000000000	0.568888888888889
6	0.932469514203152	0.171324492379170
	0.661209386466265	0.360761573048139
	0.238619186083197	0.467913934572691

La manière la plus efficace et la plus précise d'intégrer sur un intervalle donné (et borné!) un polynôme de degré 2n+1 dont on connaît les coefficients est d'utiliser une règle de Gauss-Legendre avec n+1 points.

Vrai ou Faux ?

$n+1$	$X_k, -X_{n-k}$	w_k, w_{n-k}
1	0.000000000000000	2.000000000000000
2	0.577350269189626	1.000000000000000
3	0.774596669241483	0.555555555555556
	0.000000000000000	0.888888888888889
4	0.861136311594053	0.347854845137454
	0.339981043584856	0.652145154862546
5	0.906179845938664	0.236926885056189
	0.538469310105683	0.478628670499366
	0.000000000000000	0.568888888888889
6	0.932469514203152	0.171324492379170
	0.661209386466265	0.360761573048139
	0.238619186083197	0.467913934572691

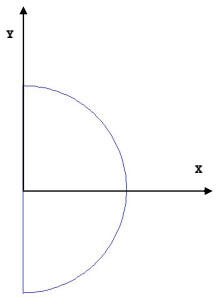
Evaluation S5 : vrai ou faux…

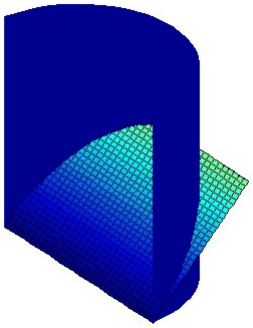
Un petit exemple en 2D…

Quelle est la valeur de l'intégrale double

$$I = \int_D x \, dx \, dy$$

où $D = \{(x,y) \in \mathbb{R}^2 \text{ tels que } x^2 + y^2 \leq 1 \text{ et } x \geq 0\}$?





D'abord les x puis les y…

$$I = \int_{-1}^{1} \left(\int_{0}^{\sqrt{1-y^2}} x \, dx \right) dy$$

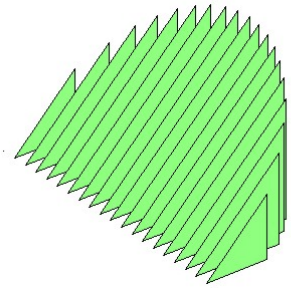
En intégrant le long de x pour chaque y

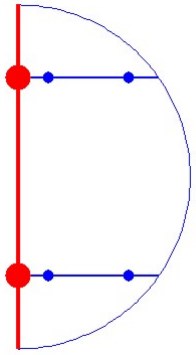
$$= \int_{-1}^{1} \left[\frac{x^2}{2} \right]_{x=0}^{x=\sqrt{1-y^2}} dy$$

$$= \int_{-1}^{1} \frac{1-y^2}{2} \, dy$$

En intégrant le long de y

$$= \left[\frac{y}{2} - \frac{y^3}{6} \right]_{y=-1}^{y=1} = 1 - \frac{1}{3} = \frac{2}{3}$$

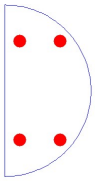




…et numériquement ?

$$I = \boxed{\text{Surface du triangle en } y = -\frac{\sqrt{3}}{3}} + \boxed{\text{Surface du triangle en } y = +\frac{\sqrt{3}}{3}}$$

$$= 2\left(\frac{1}{2}\frac{\sqrt{2}}{\sqrt{3}}\frac{\sqrt{2}}{\sqrt{3}}\right) = \frac{2}{3}$$



D'abord les y, puis les x…

$$I = \int_0^1 x \left(\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy \right) dx$$

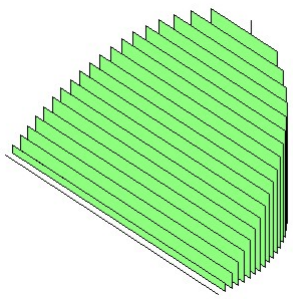
En intégrant le long de y pour chaque x

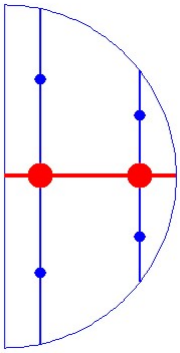
$$= \int_0^1 x \left[y \right]_{y=-\sqrt{1-x^2}}^{y=\sqrt{1-x^2}} dx$$

$$= \int_0^1 2x\sqrt{1-x^2}\, dx$$

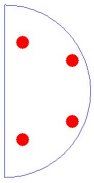
En intégrant le long de x

$$= \left[\frac{-2}{3}\left(1-x^2\right)^{3/2} \right]_{y=0}^{y=1} = \frac{2}{3}$$





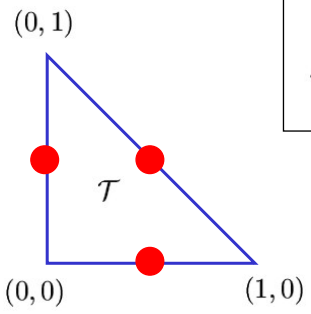
…et numériquement ?



$$I = \frac{1}{2}\left(\boxed{\text{Surface du rectangle en } x = \frac{1}{2} - \frac{\sqrt{3}}{6}} + \boxed{\text{Surface du rectangle en } x = \frac{1}{2} + \frac{\sqrt{3}}{6}} \right)$$

$$= \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)\sqrt{1 - \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)^2} + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)\sqrt{1 - \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)^2} = 0.6914 \simeq \frac{2}{3}$$

Intégration sur un triangle : Règle de Hammer à 3 points



$$\underbrace{\int_{\mathcal{T}} f(x,y)\ dx\ dy}_{I} \approx \underbrace{\sum_{k=1}^{3} w_k f(X_k, Y_k)}_{I^h}$$

	X_k	Y_k	w_k
1	0.5	0.0	1/6
2	0.5	0.5	1/6
3	0.0	0.5	1/6

Interrogation (mai 2003)

Démontrer que la formule de Hammer à trois points permet d'intégrer exactement n'importe quel polynôme à deux variables de degré deux : $a + bx + cy + dx^2 + ey^2 + fxy$

Question

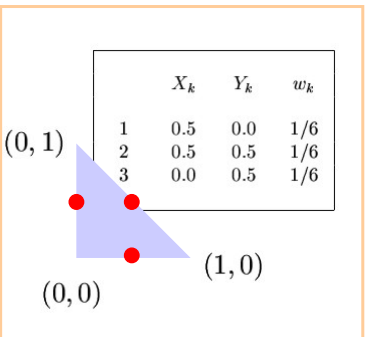
$$
\begin{aligned}
I &= \frac{a}{2} + b\int_0^1 x \int_0^{1-x} dy\, dx + c\int_0^1 y \int_0^{1-y} dx\, dy \\
&\quad + d\int_0^1 x^2 \int_0^{1-x} dy\, dx + e\int_0^1 y^2 \int_0^{1-y} dx\, dy + f\int_0^1 x \int_0^{1-x} y\, dy\, dx \\
&= \frac{a}{2} + b\left[\frac{x^2}{2} - \frac{x^3}{3}\right]_0^1 + c\left[\frac{y^2}{2} - \frac{y^3}{3}\right]_0^1 \\
&\quad + d\left[\frac{x^3}{3} - \frac{x^4}{4}\right]_0^1 + e\left[\frac{y^3}{3} - \frac{y^4}{4}\right]_0^1 + f\left[\frac{x^2}{4} - \frac{x^3}{3} + \frac{x^4}{8}\right]_0^1 \\
&= \frac{a}{2} + \frac{b}{6} + \frac{c}{6} + \frac{d}{12} + \frac{e}{12} + \frac{f}{24} \\
&= I^h
\end{aligned}
$$

□

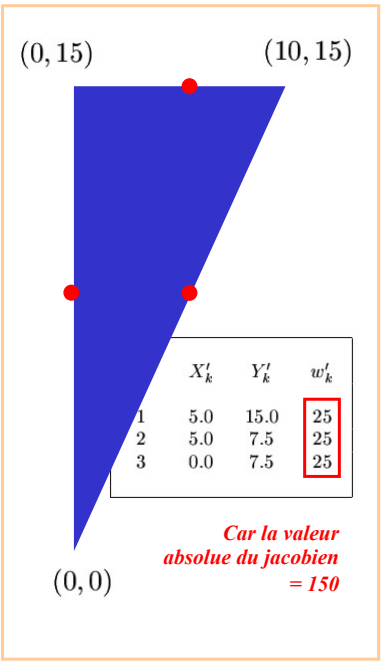
Degré de précision

Et un autre triangle ?

$$
\begin{aligned}
x' &= 10x \\
y' &= 15 - 15y
\end{aligned}
$$



	X_k	Y_k	w_k
1	0.5	0.0	1/6
2	0.5	0.5	1/6
3	0.0	0.5	1/6



	X'_k	Y'_k	w'_k
1	5.0	15.0	25
2	5.0	7.5	25
3	0.0	7.5	25

Car la valeur absolue du jacobien = 150

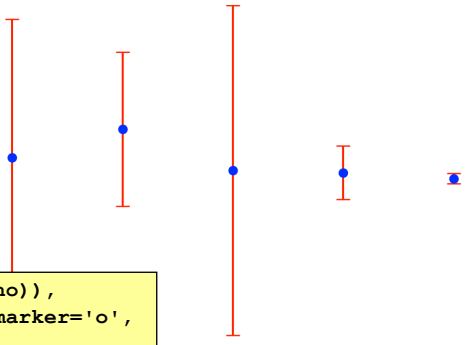
Quelques mots sur l'exploitation de mesures expérimentales…

Mesure du poids avec une estimation de l'erreur
Mesure du volume avec une estimation de l'erreur

Comment combiner au mieux ces mesures ?

```
Vol     = array([   10,   8,   8,  10,   99])
ErrVol  = array([    5,   2,   5,   1,    2])
Mass    = array([ 29.1,25.7,22.2,27.5,266.0])
ErrMass = array([  0.1, 0.1, 0.1, 0.1,  0.1])

Rho = Mass / Vol
ErrRho  = (ErrMass * Vol + ErrVol * Mass)/ (Vol*Vol)
VarRho  = ErrRho*ErrRho
```

Toutes les mesures n'ont pas la même précision…



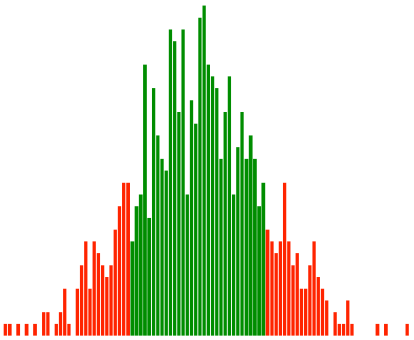
```
plt.errorbar(arange(len(Rho)),
 Rho,ErrRho,linestyle='',marker='o',
 markerfacecolor='blue',
 markeredgecolor='blue',
 color='red',capsize=5)
```

Et pourtant, la donnée la plus imprécise permettra d'améliorer la précision de la meilleure donnée, si, si !

Faisons plein de mesures virtuelles…

```
r = 500 + 15*randn(800)
plt.hist(r,100)
```

66% des mesures tombent dans l'intervalle [485,515] !



numpy.random.randn

numpy.random.randn(*d0, d1, ..., dn*)

Return a sample (or samples) from the "standard normal" distribution.

If positive, int_like or int-convertible arguments are provided, randn generates an array of shape `(d0, d1, ..., dn)`, filled with random floats sampled from a univariate "normal" (Gaussian) distribution of mean 0 and variance 1 (if any of the d_i are floats, they are first converted to integers by truncation). A single float randomly sampled from the distribution is returned if no argument is provided.

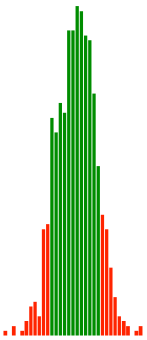
This is a convenience function. If you want an interface that takes a tuple as the first argument, use numpy.random.standard_normal instead.

Que vaut l'écart type de la moyenne de n mesures ?

66% de la moyenne de 10 mesures tombent dans l'intervalle [495,505] !

```
Effectuons 800 mesures avec un écart de 15
Moyenne                    :    499.9263731
Ecart type de la moyenne :      0.5303301
```

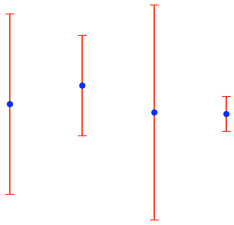
```
print('Ecart type de la moyenne : %14.7f ' 
        % (15/sqrt(800)))
```



```
r = zeros(800);
for i in range(10):
   r = r + 500 + 15*randn(800)
r = r/10
```

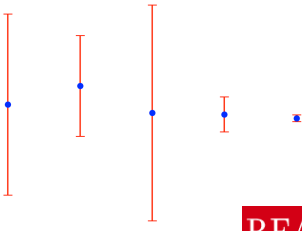
Comment mesurer au nanomètre avec une simple latte ?

```
Mesure 0 et écart type                :      2.9100000 (1.4650000)
Mesure 1 et écart type                :      3.2125000 (0.8156250)
Mesure 2 et écart type                :      2.7750000 (1.7468750)
Mesure 3 et écart type                :      2.7500000 (0.2850000)
Mesure 4 et écart type                :      2.6868687 (0.0552903)
Meilleure estimation et écart type    :      2.6918441 (0.0540956)
```

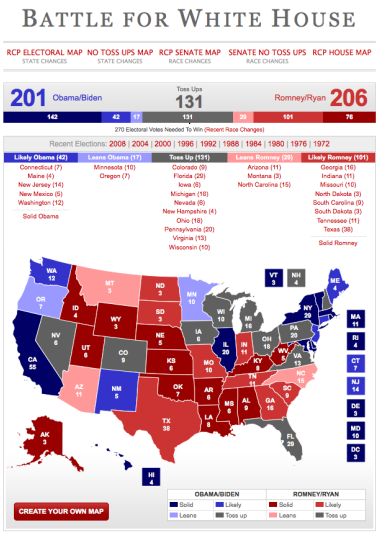


```
VarRho  = ErrRho*ErrRho
coeff = 1 / (VarRho)
coeff = coeff / sum(coeff)
BestEstimate = coeff @ Rho
VarBestEstimate = (coeff*coeff) @ VarRho
EcartBestEstimate = sqrt(VarBestEstimate)
```

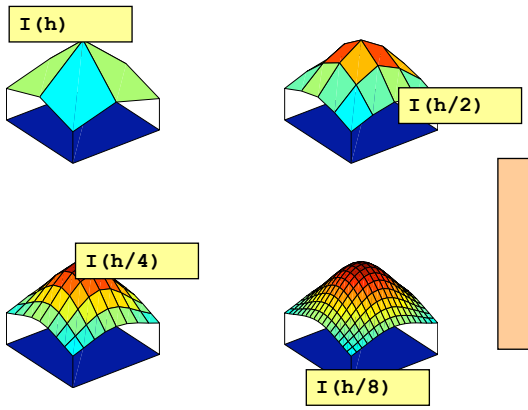
Comment combiner des sondages un peu imprécis ?



REAL CLEAR POLITICS



Pourrait-on utiliser la même idée pour les calculs numériques ?

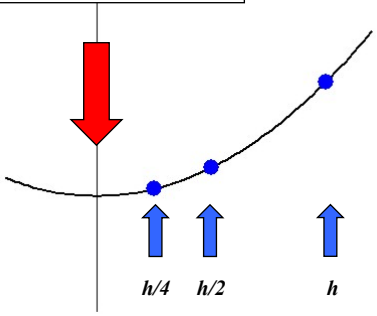


Est-il possible d'effectuer une combinaison linéaire de ces quatre valeurs numériques de l'intégrale afin d'obtenir une approximation encore plus précise que la valeur obtenue avec le plus petit pas ?

Extrapolation de Richardson

$f(\frac{h}{2^i}) \qquad i = 0, 1, 2, 3, \ldots$

Comment estimer f(0) ?



Extrapolation de Richardson

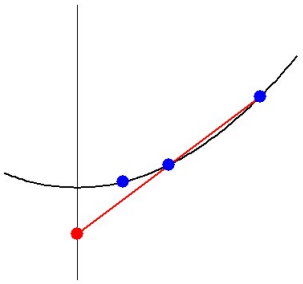
$$f(h) = f(0) + h f'(0) + \frac{h^2}{2} f''(0) + \dots$$
$$f(\tfrac{h}{2}) = f(0) + \tfrac{h}{2} f'(0) + \tfrac{h^2}{8} f''(0) + \dots$$

2 mesures
Elimination du terme du premier ordre

$f(\frac{h}{2^i}) \qquad i = 0, 1, 2, 3, \dots$

Comment estimer f(0) ?

$$f(\tfrac{h}{2}) - \tfrac{1}{2} f(h) = \left(1 - \tfrac{1}{2}\right) f(0) + \left(\tfrac{h^2}{8} - \tfrac{h^2}{4}\right) f''(0) + \dots$$
$$= \left(1 - \tfrac{1}{2}\right) f(0) - \left(1 - \tfrac{1}{2}\right) \tfrac{h^2}{4} f''(0) + \dots$$
$$\downarrow$$
$$f(0) = \frac{\left(f(\frac{h}{2}) - \frac{1}{2} f(h)\right)}{\left(1 - \frac{1}{2}\right)} + \frac{h^2}{4} f''(0) + \dots$$



Extrapolation de Richardson

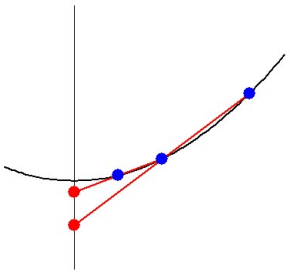
$$f(\tfrac{h}{2}) = f(0) + \tfrac{h}{2} f'(0) + \tfrac{h^2}{8} f''(0) + \dots$$
$$f(\tfrac{h}{4}) = f(0) + \tfrac{h}{4} f'(0) + \tfrac{h^2}{32} f''(0) + \dots$$

3 mesures
Elimination du terme du second ordre

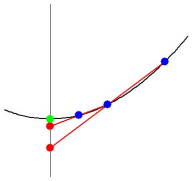
$f(\frac{h}{2^i}) \qquad i = 0, 2, 3, \dots$

Comment estimer f(0) ?

$$f(\tfrac{h}{4}) - \tfrac{1}{2} f(\tfrac{h}{2}) = \left(1 - \tfrac{1}{2}\right) f(0) + \left(\tfrac{h^2}{32} - \tfrac{h^2}{16}\right) f''(0) + \dots$$
$$= \left(1 - \tfrac{1}{2}\right) f(0) - \left(1 - \tfrac{1}{2}\right) \tfrac{h^2}{16} f''(0) + \dots$$
$$\downarrow$$
$$f(0) = \frac{\left(f(\frac{h}{4}) - \frac{1}{2}(f(\frac{h}{2}))\right)}{\left(1 - \frac{1}{2}\right)} + \frac{h^2}{16} f''(0) + \dots$$



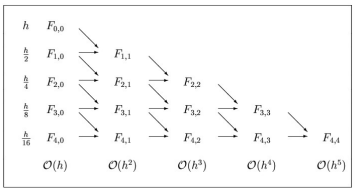
… et finalement



$$f(0) = \frac{\left(\frac{\left(f(\frac{h}{4}) - \frac{1}{2}f(\frac{h}{2})\right)}{\left(1-\frac{1}{2}\right)}\right) - \frac{1}{4}\left(\frac{\left(f(\frac{h}{2}) - \frac{1}{2}f(h)\right)}{\left(1-\frac{1}{2}\right)}\right)}{\left(1-\frac{1}{4}\right)} + \frac{h^3}{48}f'''(0) + \ldots$$

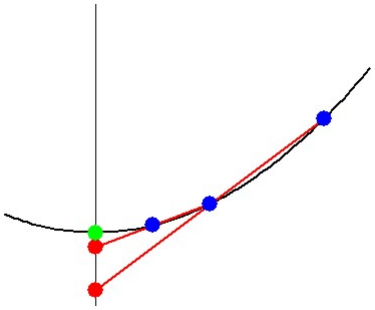
$$F_{i,k} = \frac{\left(F_{i,k-1} - \frac{1}{2^k}F_{i-1,k-1}\right)}{\left(1-\frac{1}{2^k}\right)}$$

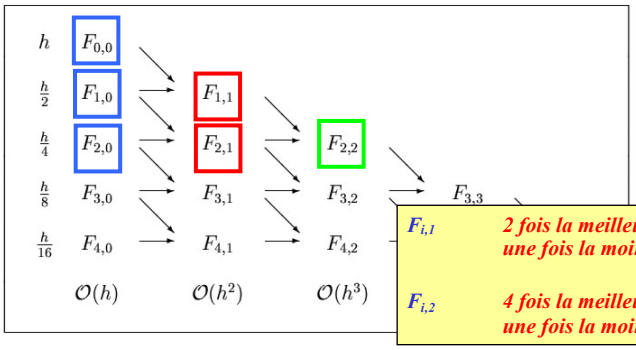
Disposition pratique des calculs



$F_{i,1}$	*2 fois la meilleure estimation moins une fois la moins bonne*
$F_{i,2}$	*4 fois la meilleure estimation moins une fois la moins bonne le tout divisé par 3*

Extrapolation de Richardson





$F_{i,1}$	*2 fois la meilleure estimation moins une fois la moins bonne*
$F_{i,2}$	*4 fois la meilleure estimation moins une fois la moins bonne le tout divisé par 3*

Romberg = application de l'extrapolation de Richardson à la méthode composite des trapèzes avec des pas décroissants

$$
\begin{aligned}
I^{h} &= \tfrac{h}{2}\,(U_0 + U_1) \\
I^{h/2} &= \tfrac{h}{4}\,(U_0 + 2U_1 + U_2) \\
I^{h/4} &= \tfrac{h}{8}\,(U_0 + 2U_1 + 2U_2 + 2U_3 + U_4) \\
I^{h/8} &= \tfrac{h}{16}\,(U_0 + 2U_1 + 2U_2 + 2U_3 + \ldots + 2U_7 + U_8) \\
I^{h/16} &= \tfrac{h}{32}\,(U_0 + 2U_1 + 2U_2 + 2U_3 + \ldots + 2U_{15} + U_{16})
\end{aligned}
$$

Méthode composite de trapèzes
h = (b-a) pour toutes les expressions

Extrapolation de Richardson

Estimer $I(0)$ sachant que

$$
\begin{aligned}
I(h) &= I^{h} \\
I(h/2) &= I^{h/2} \\
I(h/4) &= I^{h/4} \\
I(h/8) &= I^{h/8} \\
I(h/16) &= I^{h/16}
\end{aligned}
$$

… et pratiquement

$$
I_{i,k} = \frac{\left(I_{i,k-1} - \frac{1}{2^{2k}} I_{i-1,k-1}\right)}{\left(1 - \frac{1}{2^{2k}}\right)}
$$

Le terme d'erreur ne fait intervenir que des puissances paires….

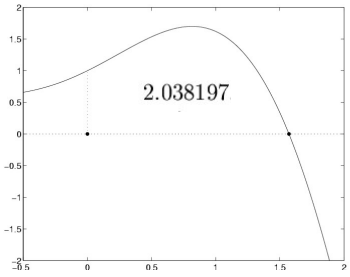
$I_{i,1}$ *4 fois la meilleure estimation moins une fois la moins bonne le tout divisé par 3*

h	$I_{0,0}$			
$\frac{h}{2}$	$I_{1,0}$	$I_{1,1}$		
$\frac{h}{4}$	$I_{2,0}$	$I_{2,1}$	$I_{2,2}$	
$\frac{h}{8}$	$I_{3,0}$	$I_{3,1}$	$I_{3,2}$	$I_{3,3}$
	$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\mathcal{O}(h^8)$

Simpson *Boole*

Exemple

$$I = \int_0^{\pi/2} (x^2 + x + 1)\cos x \, dx$$



i	$I_{i,0}$	$I_{i,1}$	$I_{i,2}$	$I_{i,3}$	$I_{i,4}$
0	0.785398				
1	1.726813	2.040617			
2	1.960534	2.038441	2.038296		
3	2.018794	2.038214	2.038199	2.038197	
4	2.033347	2.038198	2.038197	2.038197	2.038197
	$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\mathcal{O}(h^8)$	$\mathcal{O}(h^{10})$

Romberg : Implémentation récursive ?

```
h=1;d=0
Iq=Integrale exacte
I=Iq-1
while (Iq-I)~=0 :
  I=romberg(d,d,h)
  d=d+1;
```

Mon groupe favori…. lundi à 22h15.

« Afin de limiter les boucles trop ennuyeuses nous avons utilisé fonction interne. Mais nous ne nous sommes rendus compte que trop tard qu'une telle fonction ne peut s'appeler elle-même, et c'est ce qui, nous le présumons, engendre une erreur lors de l'exécution du programme…. »

Romberg

```
h=1;d=0
Iq=Integrale exacte
I=Iq-1
while (Iq-I)~=0 :
  I=romberg(d,d,h)
  d=d+1;
```

```
def romberg(i,k,h):
  if k==0 :
    Integration par les trapezes…
  else :
    I=(romberg(i,k-1,h)-
       (romberg(i-1,k-1,h)/(2**(2*k))))/(1-(1/(2**(2*k))))
  return I
```

Implémentation récursive de la méthode de Romberg
Ecriture très compacte
Efficacité très faible

Quelques moments plus tard…

```
h=1;d=0
Iq=Integrale exacte
I=Iq-1
while (Iq-I)!=0 :
  I=romberg(d,d,h)
  d=d+1;
```

Mon groupe favori, mardi à 19h52

X propos de la monstuositX …
HUm .
Bien , donc c vrmnt très nul , dixit ma soeur is my clock , suffit de remplacer par while ((Iq - I) != 0)

Voilà on est nul

Et le matin…

Mon groupe favori, mercredi à 01h07

« Voici en annexe la version finale. Il a été amélioré et fonctionne à présent bien. »

Mon groupe favori, mercredi à 09h

« Voici en annexe la version finale. Il a été amélioré et fonctionne à présent bien. Ce programme fut fait dans son intégrité par notre groupe 1183. Toute similitude avec un autre programme déjà existant serait purement fortuite... Il nous a semblé que le mail envoyé hier contenait peut-être des fichiers vides suite à une erreur du serveur mail ; nous nous permettons donc de

La version 3 du programme…

```
d=0
p=1/1000
I=0
I_old=0
while ( round(I/p) < round(I_old/p) or I_old==0 ):
  I_old=I
  I=romberg(f,d,d,0,1)
  d=d+1
```

On n'est jamais sûr que ce test sera satisfait…
Possibilité réelle de programme ne s'arrêtant jamais jamais jamais

La récursivité avec Romberg, c'est aussi une…

```
h=1;d=0
Iq = Integrale exacte
I = Iq-1
while (d!=6):
    I=romberg(d,d,h)
    d=d+1
```

```
def romberg(i,k,h):
  if k==0 :
    Integration par les trapezes…
  else :
    I=(romberg(i,k-1,h)-
       (romberg(i-1,k-1,h)/(2**(2*k))))/(1-(1/(2**(2*k))))
    print(' Etape %i %i -> I  %21.14e' % (i,k,I))
  return I
```

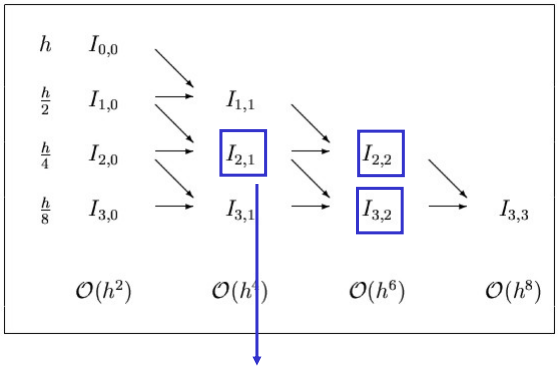
Beaucoup de calculs répétés inutilement…

```
 >>>>
I =
    1.3244
 Romberg : etape 1 1 -> I = 1.19688584716124e+000
I =
    1.1969
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 1 1 -> I = 1.19688584716124e+000
 Romberg : etape 2 2 -> I = 1.19497479612220e+000
I =
    1.1950
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 3 2 -> I = 1.19495798442620e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 1 1 -> I = 1.19688584716124e+000
 Romberg : etape 2 2 -> I = 1.19497479612220e+000
 Romberg : etape 3 3 -> I = 1.19495771757388e+000
I =
    1.1950
```

```
 Romberg : etape 4 1 -> I = 1.19495821928187e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 4 2 -> I = 1.19495766722064e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 3 2 -> I = 1.19495798442620e+000
 Romberg : etape 4 3 -> I = 1.19495766218563e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 3 2 -> I = 1.19495798442620e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 1 1 -> I = 1.19688584716124e+000
 Romberg : etape 2 2 -> I = 1.19497479612220e+000
 Romberg : etape 3 3 -> I = 1.19495771757388e+000
 Romberg : etape 4 4 -> I = 1.19495766196842e+000
I =
    1.1950
```

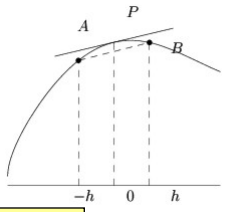
```
 Romberg : etape 5 1 -> I = 1.19495769682481e+000
 Romberg : etape 4 1 -> I = 1.19495821928187e+000
 Romberg : etape 5 2 -> I = 1.19495766199434e+000
 Romberg : etape 4 1 -> I = 1.19495821928187e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 4 2 -> I = 1.19495766722064e+000
 Romberg : etape 5 3 -> I = 1.19495766191138e+000
 Romberg : etape 4 1 -> I = 1.19495821928187e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 4 2 -> I = 1.19495766722064e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 3 2 -> I = 1.19495798442620e+000
 Romberg : etape 4 3 -> I = 1.19495766218563e+000
 Romberg : etape 5 4 -> I = 1.19495766191030e+000
 Romberg : etape 4 1 -> I = 1.19495821928187e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 4 2 -> I = 1.19495766722064e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 3 2 -> I = 1.19495798442620e+000
 Romberg : etape 4 3 -> I = 1.19495766218563e+000
 Romberg : etape 3 1 -> I = 1.19496650020032e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 3 2 -> I = 1.19495798442620e+000
 Romberg : etape 2 1 -> I = 1.19509423681214e+000
 Romberg : etape 1 1 -> I = 1.19688584716124e+000
 Romberg : etape 2 2 -> I = 1.19497479612220e+000
 Romberg : etape 3 3 -> I = 1.19495771757388e+000
 Romberg : etape 4 4 -> I = 1.19495766196842e+000
 Romberg : etape 5 5 -> I = 1.19495766191025e+000
I =
    1.1950
>>
```

Il est beaucoup plus efficace de suivre le tableau des calculs….



Est nécessaire pour I_{22} et I_{32}

Plan du cours de méthodes numériques



Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non linéaires ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

$$D = u'(x) = \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}$$

$$D^{h_k} = \frac{u(x+h_k) - u(x)}{h_k} \qquad k = 1, 2, \ldots, n, \ldots$$

Dérivation numérique…

Méthodes conceptuellement simples…
- Différences finies d'ordre n
- Différences finies centrées et unilatérales

… mais numériquement instables
- Erreurs d'arrondi
- Concept de stabilité numérique

Valeur exacte

$$E^h = D - D^h$$

Approximation numérique

Une petite expérience numérique….

h_k	$\exp(1+h_k)$	D^{h_k}	E^{h_k}
$h_0 = 1$	7.389056099	4.670774270	-1.952492442
$h_1 = 0.1$	3.004166024	2.858841955	-0.140560126
$h_2 = 0.01$	2.745601015	2.731918656	-0.013636827
$h_3 = 0.001$	2.721001470	2.719641423	-0.001359594
$h_4 = 0.0001$	2.718553670	2.718417747	-0.000135919
$h_5 = 0.00001$	2.718309011	2.718295420	-0.000013591
$h_6 = 10^{-6}$	2.718284547	2.718283187	-0.000001359
$h_7 = 10^{-7}$	2.718282100	2.718281964	-0.000000140
$h_8 = 10^{-8}$	2.718281856	2.718281777	0.000000007
$h_9 = 10^{-9}$	2.718281831	2.718281600	-0.000000660
$h_{10} = 10^{-10}$	2.718281829	2.718278935	-0.000001548
$h_{11} = 10^{-11}$	2.718281828	2.718270053	-0.000032634
$h_{12} = 10^{-12}$	2.718281828	2.718270053	-0.000432314
$h_{13} = 10^{-13}$	2.718281828	2.713385072	0.000455864
$h_{14} = 10^{-14}$	2.718281828	2.664535259	-0.035071273
$h_{15} = 10^{-15}$	2.718281828	2.664535259	-0.390342640
$h_{16} = 10^{-16}$	2.718281828	0.000000000	2.718281828
$h_{17} = 10^{-17}$	2.718281828	0.000000000	2.718281828
$h_{18} = 10^{-18}$	2.718281828	0.000000000	2.718281828
$h_{19} = 10^{-19}$	2.718281828	0.000000000	2.718281828

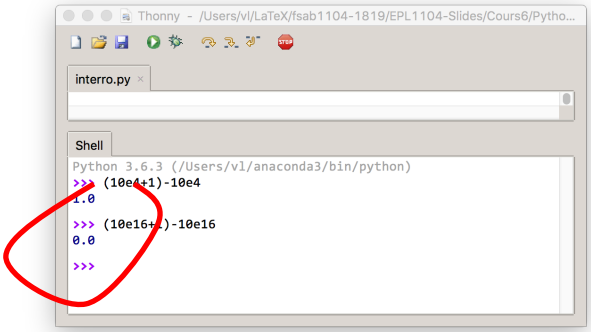
***Théoriquement**, la méthode avec une arithmétique exacte devrait converger vers la valeur exacte*

***Pratiquement avec Matlab**, l'erreur diminue et puis recommence à augmenter...*

Il s'agit d'une catastrophe numérique !

Erreurs d'arrondi : c'est quoi ?

Un exemple très simple qui montre l'importance des erreurs d'arrondi



Arithmétique à précision finie

Valeur exacte

Nombre de chiffres significatifs

$$\left|\frac{x-\tilde{x}}{x}\right| < \frac{1}{2}\,10^{-d}$$

Représentation dans l'ordinateur

Calculateur avec 4 chiffres pour représenter un nombre :

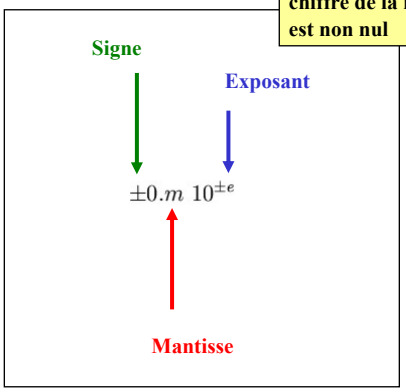
Effectuons l'opération	**1.348 + 9.999**
Valeur exacte	**11.347**
Valeur représentée par le calculateur	**11.35**

Comment utiliser astucieusement la mémoire disponible ?

x	$\tilde{x}$	nombre de chiffres significatifs : d
1.000001	0.999998	$\left\|\frac{x-\tilde{x}}{x}\right\| \approx 0.000003 < \frac{1}{2}\,10^{-5}$ — 5
0.001234	0.001231	$\left\|\frac{x-\tilde{x}}{x}\right\| \approx 0.002 < \frac{1}{2}\,10^{-2}$ — 2
0.000013	0.000009	$\left\|\frac{x-\tilde{x}}{x}\right\| \approx 0.31 < \frac{1}{2}$ — 0

$$
\begin{array}{rcrcrcl}
 & & & & & & \overbrace{}^{11\text{ zéros}} \\
0.5000 & 10^{12} & = & 5.000 & 10^{11} & = & 5\,00000000000\,.000000000000000 \\
-0.3275 & 10^{04} & = & -3.275 & 10^{03} & = & -3275.000000000000000 \\
0.5723 & 10^{01} & = & 5.723 & 10^{00} & = & 5.723000000000000 \\
0.9422 & 10^{-11} & = & 9.422 & 10^{-12} & = & 0.\underbrace{00000000000}_{11\text{ zéros}}9422
\end{array}
$$

Représentation normalisée : le premier chiffre de la mantisse est non nul



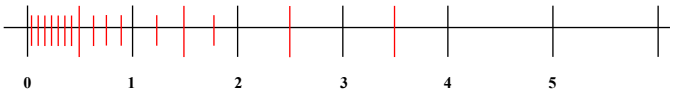
Arithmétique en virgule flottante

ou l'utilisation astucieuse de la mémoire disponible...

Base binaire

IEEE single (32bits)	**mantisse**	**23+1 bits**
	exposant	**8 bits**
	limites (range)	**10^{-38} 10^{+38}**
	arrondi à l'unité	**2^{-24} (approximativement 10^{-8})**
IEEE double (64 bits)	**mantisse**	**52+1 bits**
	exposant	**11 bits**
	limites (range)	**10^{-308} 10^{+308}**
	arrondi à l'unité	**2^{-53} (approximativement 10^{-16})**

En réalité, un ordinateur utilise des nombres binaires…



Il est important de réaliser que les nombres représentables ne sont pas également espacés

Exemple représenté : mantisse de 3 bits, exposant compris entre –1 et 3.

Calcul en simple *(32bits)* ou double précision *(64bits)*

Python : l'instruction du jour…

```
>>> import numpy as np
>>> np.finfo(float).eps
2.220446049250313e-16
>>> np.finfo(float).min
-1.7976931348623157e+308
>>> np.finfo(float).max
1.7976931348623157e+308
>>> np.finfo(float).tiny
2.2250738585072014e-308
```

realmax = 1.7977e+308
realmin = 2.22507^{e}-308

Il y a beaucoup plus de nombres représentables près de x_{min} que près de x_{max}

x_{max} = 1.7976931348623158e+308
$\underline{x}_{max}$ = 1.7976931348623157e+308

$x_{max} - \underline{x}_{max}$ est de l'ordre de 10^{292}

x_{min} = 2.2250738507201e-308
$\underline{x}_{min}$ = 2.2250738507202e-308

$\underline{x}_{min} - x_{min}$ est de l'ordre de 10^{-323}

Les dangers de la soustraction…

ou pourquoi les formules de différences sont numériquement instables.

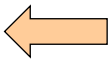
Représentation avec 12 chiffres	
Effectuons l'opération x-y	
Valeur exacte de x	**3.1234567845578**
Valeur de x dans l'ordinateur	**3.12345678456**
Valeur exacte de y	**3.1234567844660**
Valeur de y dans l'ordinateur	**3.12345678447**
Résultat exact	**0.0000000000918**
Valeur approchée	**0.00000000009**

12 chiffres significatifs pour les données

1 chiffre significatif pour le résultat

Les facéties du calcul en virgule flottante…

```
>>> a=1; b=1
>>> while (a+b) != a:
...   b=b/2
...
>>> print(b)
1.1102230246251565e-16
```

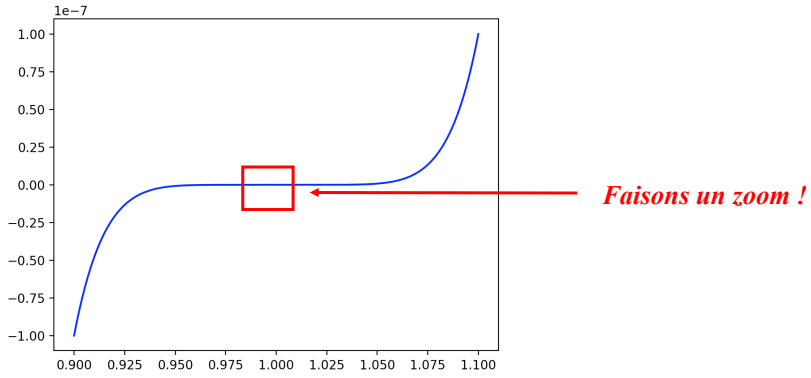


Ce programme ne devrait jamais se terminer dans l'espace des nombres réels…
Mais il se termine très très rapidement dans l'espace des nombres représentables par `python`

Les règles de l'associativité et de la distributivité sont violées !

```
>>> a=1.0e+308; b=1.1e+308; c=-1.001e+308
>>> print(a+(b+c))
1.099e+308
>>> print((a+b)+c)
inf
>>>
```

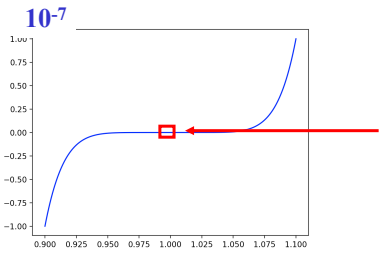
Les graphes… le meilleur et le pire !

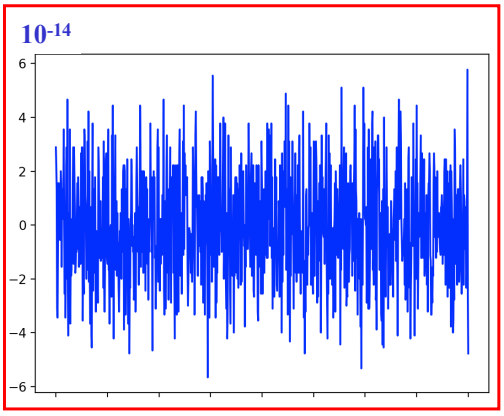



```
p = [1,-7,21,-35,35,-21,7,-1]
x = linspace(0.9,1.1,1000)
plt.plot(x,polyval(p,x),'-b')
```

Et voilà un résultat débile :-)

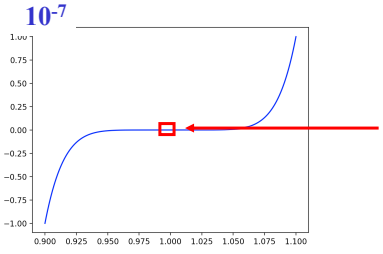
```
p = [1,-7,21,-35,35,-21,7,-1]
x = linspace(1-2e-4,1+2e-4,1000)
plt.plot(x,polyval(p,x),'-b')
```

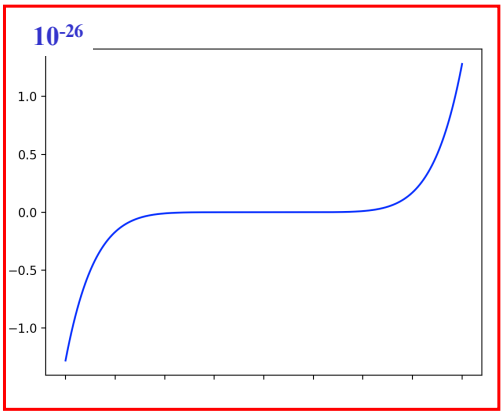




Et le même graphe correct :-)

```
x = linspace(1-2e-4,1+2e-4,1000)
plt.plot(x,power((x-1.0),7),'-b')
```





Propagation des erreurs d'arrondi pour les différences

$$u'(0) = \underbrace{\frac{(U_h - U_{-h})}{2h}}_{D^h} + \underbrace{(-1)\frac{h^2}{6}u^{(3)}(\xi)}_{E^h}$$

Erreur de discrétisation

Erreurs d'arrondi sur les données

$$U_h = \tilde{U}_h + \tilde{e}_h$$

$$U_{-h} = \tilde{U}_{-h} + \tilde{e}_{-h}$$

Pratiquement…

Erreur de discrétisation

Propagation des erreurs d'arrondi

$$u'(0) = \frac{(U_h - U_{-h})}{2h} + (-1)\frac{h^2}{6}u^{(3)}(\xi)$$

$$u'(0) = \underbrace{\underbrace{\frac{(\tilde{U}_h - \tilde{U}_{-h})}{2h}}_{\tilde{D}^h} + \frac{(\tilde{e}_h - \tilde{e}_{-h})}{2h}}_{D^h} + \underbrace{(-1)\frac{h^2}{6}u^{(3)}(\xi)}_{E^h}$$

$\tilde{E}^h$

On néglige l'erreur d'arrondi commise sur les opérations pour calculer la différence, car elle est peu importante par rapport à l'erreur propagée

Erreur totale

Estimation du pas optimal

Calcul de h qui minimise l'erreur totale en valeur absolue

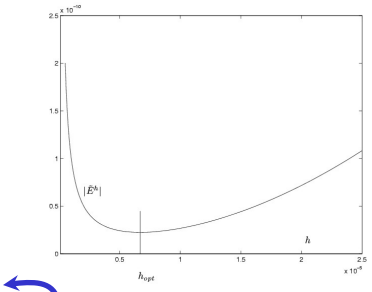
$$|\tilde{E}^h| \leq \frac{\epsilon}{h} + \frac{C\,h^2}{6}$$

$$h = \left(\frac{3\,\epsilon}{C}\right)^{1/3}$$

Exemple

calcul de la dérivée de cos(0.7)

h	$\tilde{D}^h$	$\tilde{E}^h$
1	-0.54209049171057	0.10212719552713
0.1	-0.64314452781256	0.00107315942513
0.01	-0.64420695032992	0.00001073690777
0.001	-0.64421757986810	0.00000010736959
0.0001	-0.64421768616374	0.00000000107395
1.0000000e-005	-0.64421768722345	0.00000000001424
1.0000000e-006	-0.64421768725120	-0.00000000001351
1.0000000e-007	-0.64421768697365	0.00000000026404
1.0000000e-008	-0.64421769030432	-0.00000000306663
1.0000000e-009	-0.64421767920209	0.00000000803561
1.0000000e-010	-0.64421745715748	0.00000023008021
1.0000000e-011	-0.64421801226899	-0.00000032503130
1.0000000e-012	-0.64420691003875	0.00001077719894
1.0000000e-013	-0.64448446579490	-0.00026677855721
1.0000000e-014	-0.64392935428259	0.00028833295510
1.0000000e-015	-0.61062266354384	0.03359502369385
1.0000000e-016	-0.55511151231258	0.08910617492511
1.0000000e-017	0.00000000000000	0.64421768723769
1.0000000e-018	0.00000000000000	0.64421768723769
1.0000000e-019	0.00000000000000	0.64421768723769



Application de l'extrapolation de Richardson à des différences centrées avec des pas décroissants

$$u'(0) = \frac{(U_h - U_{-h})}{2h} - \frac{h^2}{6}u^{(3)}(\xi)$$

Différences centrées h=0.1

Estimer $D(0)$ sachant que

$$\begin{aligned} D(h) &= -0.64314452781256 \\ D(h/10) &= -0.64420695032992 \\ D(h/100) &= -0.64421757986810 \\ D(h/1000) &= -0.64421768616374 \end{aligned}$$

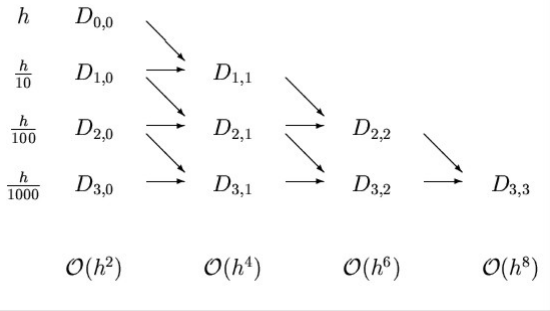
Extrapolation de Richardson

… et pratiquement

$$D_{i,k} = \frac{\left(D_{i,k-1} - \frac{1}{10^{2k}} D_{i-1,k-1}\right)}{\left(1 - \frac{1}{10^{2k}}\right)}$$

Le terme d'erreur ne fait intervenir que des puissances paires….

$D_{i,1}$ 100 fois la meilleure estimation moins une fois la moins bonne le tout divisé par 99



Exemple

h=0.1, h/10, h/100, h/1000

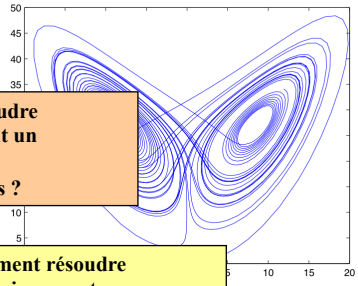
i	$D_{i,0}$	$D_{i,1}$	$D_{i,2}$	$D_{i,3}$
0	-0.64314452781256			
1	-0.64420695032992	-0.64421768187050		
2	-0.64421757986810	-0.64421768723717	-0.64421768723771	
3	-0.64421768616374	-0.64421768723743	-0.64421768723743	-0.64421768723743
	$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\mathcal{O}(h^8)$

Inutile, pourquoi ?

h=0.1, h/2, h/4, h/8

i	$D_{i,0}$	$D_{i,1}$	$D_{i,2}$	$D_{i,3}$
0	-0.64314452781256			
1	-0.64394929675235	-0.64421755306561		
2	-0.64415058332564	-0.64421767885006	-0.64421768723569	
3	-0.64420091086648	-0.64421768723743	-0.64421768723765	-0.64421768723766
	$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\mathcal{O}(h^8)$

Plan du cours de méthodes numériques



Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non linéaires ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

1

Trouver $u(x)$ tel que

$$\begin{cases} u'(x) &= f(x, u(x)), & x \in [a, b] \\ u(a) &= \bar{u} \end{cases}$$

Applications :
très nombreuses dans tous les domaines

Problème de Cauchy

Valeur exacte

$$u(X_i) \approx u^h(X_i) = U_i$$

Approximation numérique

Questions théoriques
- **Existence, unicité et régularité d'une solution**
- **Stabilité d'une équation différentielle**

Méthodes numériques
- **Stabilité d'une méthode**
- **Précision d'une méthode**

2

Savez-vous si…

Problème linéaire ou non linéaire ?

Est-ce que f est une fonction linéaire de u ?

Est-ce que f est une fonction linéaire de x ?

Problème homogène ou non homogène ?

Dépendance explicite ou non de f par rapport à x ?

Solution particulière et solution du problème homogène

Problème scalaire ou vectoriel ?

***u* scalaire ou *u* vecteur ?**

***x* scalaire : équation différentielle ordinaire (*x* = le temps très souvent)**

***x* vecteur : équation aux dérivées partielles (CM10-CM11-CM12)**

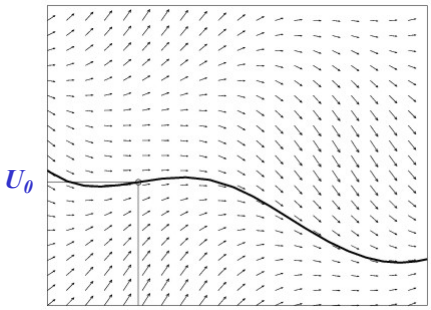
Ordre d'une équation différentielle ?

problème scalaire d'ordre *n* = système de *n* équations d'ordre un

Nous allons juste construire des méthodes pour résoudre des systèmes d'équations d'ordre 1

3

Interprétation graphique



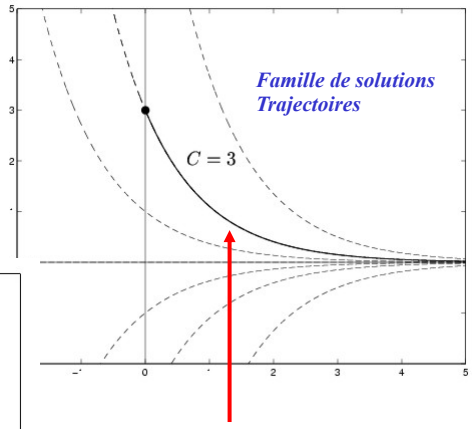
Résoudre $u'(x) = \sin(x) + \cos(u(x))$
avec $u(X_0) = U_0$

est équivalent à

Construire une courbe
qui passe par (X_0, U_0)
qui a une pente en tout point x qui vaut sin(x) + cox(u(x))

4

u'(x) = -u(x) est une équation différentielle stable…



Trouver $u(x)$ tel que

$$\begin{cases} u'(x) &= -u(x), & x \in [a,b] \\ u(a) &= \bar{u} \end{cases}$$

$$u(x) = \bar{u}\ e^{-(x-a)}$$

Lorsque les solutions se rejoignent lorsque x tend vers l'infini, on dit que le problème différentiel est stable ou bien posé

5

Sensibilité à une perturbation de la condition initiale

Problème non perturbé

Trouver $u(x)$ tel que

$$\begin{cases} u'(x) &= -u(x), & x \in [a,b] \\ u(a) &= \bar{u} \end{cases}$$

Trouver $u_\epsilon(x)$ tel que

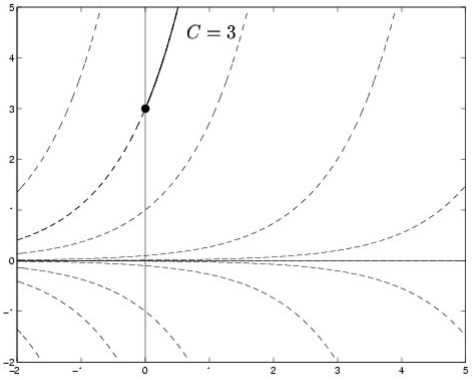
$$\begin{cases} u'_\epsilon(x) &= -u_\epsilon(x), & x \in [a,b] \\ u_\epsilon(a) &= \bar{u} + \epsilon \end{cases}$$

Problème perturbé

$$\underbrace{u_\epsilon(x) - u(x)}_{e_\epsilon(x)} = \epsilon\ e^{-(x-a)}$$

L'écart entre la solution du problème perturbé et la solution du problème non pertubé diminue progressivement de manière exponentielle...

6



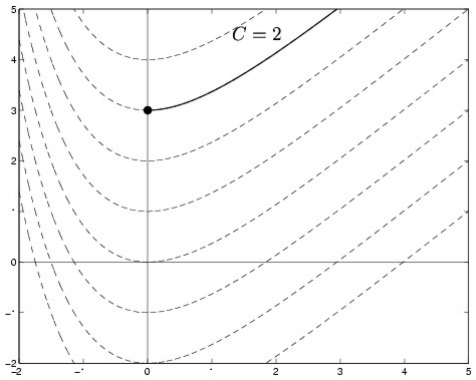
$$u(x) = \bar{u}\ e^{(x-a)}$$

u'(x) = u(x)
est une équation
différentielle instable.

Si la solution analytique est déjà instable, il semble illusoire de croire que la solution discrète sera stable par rapport aux erreurs d'arrondi !

7

u'(x) = 1-e^{-x}
n'est
ni stable,
ni instable.



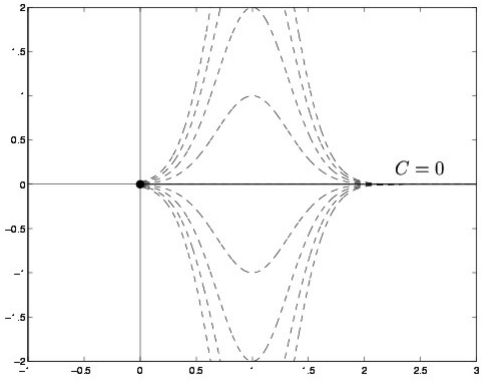
$$u(x) = C + x + e^{-x}$$

L'écart dû à une perturbation reste constant

Au bénéfice du doute, on dit classiquement que le problème est encore stable

8

u'(x) = -10(x-1)u(x)
est un peu stable
et un peu instable…



$$u(x) = Ce^{-5(x-1)^2}$$

9

Comment savoir dans le cas général ?

Idée

Presque toute l'information locale est contenue dans le *jacobien*.

Equation différentielle pour la différence entre solution du problème perturbé et solution du problème non-perturbé

$$e'_\epsilon(x) = f(x, u_\epsilon(x)) - f(x, u(x))$$

En effectuant un développement de Taylor de $f(x, v)$ pour la seconde variable v autour du point $(x, u(x))$,

$$\approx f(x, u(x)) + \left.\frac{\partial f}{\partial u}\right|_{(x,u)=(x,u(x))} \Big(u_\epsilon(x) - u(x)\Big) - f(x, u(x))$$

$$\approx \left.\frac{\partial f}{\partial u}\right|_{(x,u)=(x,u(x))} e_\epsilon(x)$$

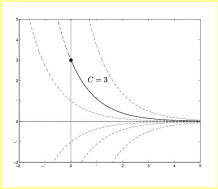
$$J(x, u(x)) = \left.\frac{\partial f}{\partial u}\right|_{(x,u(x))}$$

jacobien

10

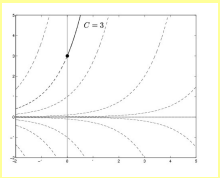
Bilan

u'(x) = -u(x)



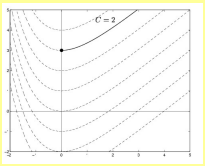
J = -1 ***stable***

u'(x) = u(x)



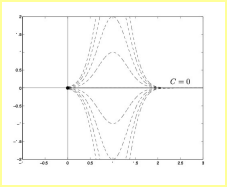
J = 1 ***instable***

u'(x) = f(x)



J = 0 ***écart constant***

u'(x) = -10(x-1)u(x)



J > 0 si x<1 ***instable***
J < 0 si x>1 ***stable***

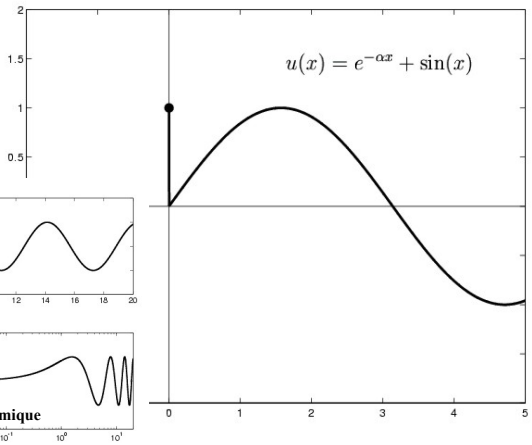
11

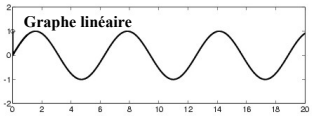
Trouver $u(x)$ tel que

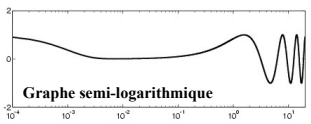
$$\begin{cases} u'(x) &= -\alpha\Big(u(x) - \sin(x)\Big) + \cos(x), & x \in [0,5] \\ u(0) &= 1 \end{cases}$$

Très difficile à résoudre
pour beaucoup de méthodes numériques

Problème stable mais raide



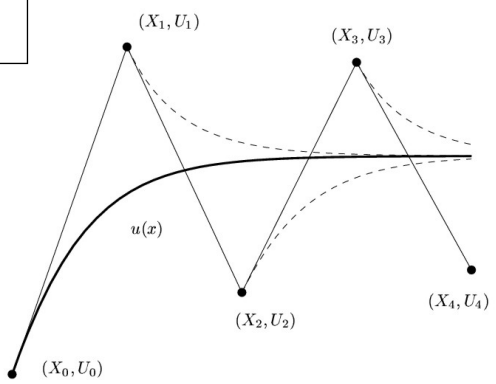




12

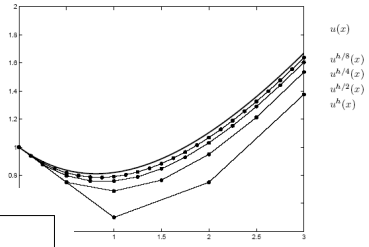
Méthode d'Euler explicite

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$



13

Euler explicite…



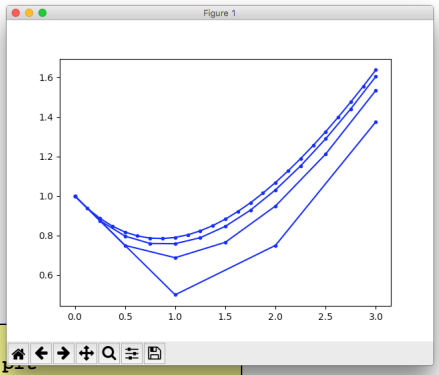
Méthode d'Euler explicite pour $u' = (x - u)/2$

X_i	$u^h(X_i)$	$u^{h/2}(X_i)$	$u^{h/4}(X_i)$	$u^{h/8}(X_i)$	$u(X_i)$
0.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.1250				0.9375	0.9432
0.2500			0.8750	0.8867	0.8975
0.3750				0.8469	0.8621
0.5000		0.7500	0.7969	0.8174	0.8364
0.7500			0.7598	0.7868	0.8119
1.0000	0.5000	0.6875	0.7585	0.7902	0.8196
2.0000	0.7500	0.9492	1.0308	1.0682	1.1036
3.0000	1.3750	1.5339	1.6043	1.6374	1.6694

… converge

14

Euler explicite u'= (x-u)/2

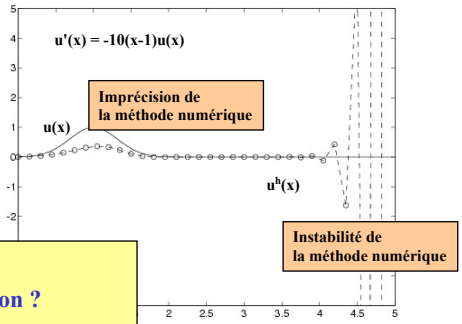



```
from numpy import *
from matplotlib import pyplot as plt

Xstart = 0; Xend = 3; Ustart = 1;
for n in [3,6,12,24]:
  h = (Xend-Xstart)/n
  X = linspace(Xstart,Xend,n+1)
  U = zeros(n+1); U[0] = Ustart
  for i in range(n):
    U[i+1] =  U[i] + h*(X[i]-U[i])/2
  plt.plot(X,U,'.-b')
plt.show()
```

15

Et parfois, cela ne marche pas !

$$X_i = a + hi \quad i = 0, \ldots, m,$$



Définir m points

- Comment choisir le pas de discrétisation ?
- Pas constant ou adaptatif ?

Chercher une approximation U_k en X_k

- Comment avoir une méthode stable ?
- Comment avoir une méthode précise ?

$$u(X_i) \approx u^h(X_i) = U_i$$

16

Méthodes explicites de Taylor :

Effectuons un développement de Taylor...

$$u(x) = u(X_0) + (x - X_0)\, u'(X_0) + \frac{(x - X_0)^2}{2!}\, u''(X_0) + \cdots$$

En vertu du problème à la valeur initiale,

$$= U_0 + (x - X_0)\, \widehat{f}(X_0) + \frac{(x - X_0)^2}{2!}\, \widehat{f}'(X_0) + \cdots$$

En tenant compte que $\widehat{f}' = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} u'$ et que $u' = \widehat{f}$

$$= U_0 + (x - X_0)\, f(X_0, U_0) + \frac{(x - X_0)^2}{2!} \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} f \right)_{(X_0, U_0)} + \frac{(x - X_0)^3}{3!} \left(\frac{\partial^2 f}{\partial x^2} + 2 \frac{\partial^2 f}{\partial x \partial u} f + \frac{\partial^2 f}{\partial u^2} f^2 + \frac{\partial f}{\partial x} \frac{\partial f}{\partial u} + \left(\frac{\partial f}{\partial u} \right)^2 f \right)_{(X_0, U_0)} + \cdots$$

$$\widehat{f} : x \to f(x, u(x))$$

Attention, la fonction à une variable ressemble à la fonction à deux variables, mais ce n'est pas la même chose !!!

17

Méthodes de Taylor d'ordre n

$$U_{i+1} = U_i + h \underbrace{\left[f + \frac{h}{2!} \left. \frac{df}{dx} \right|_{u'=f} + \cdots + \frac{h^{n-1}}{n!} \left. \frac{d^{(n-1)} f}{dx^{n-1}} \right|_{u'=f} \right]_{(X_i, U_i)}}_{\Phi(X_i, U_i)}$$

$$\left. \frac{d}{dx} \right|_{u'=f} = \left(\frac{\partial}{\partial x} + f \frac{\partial}{\partial u} \right)$$

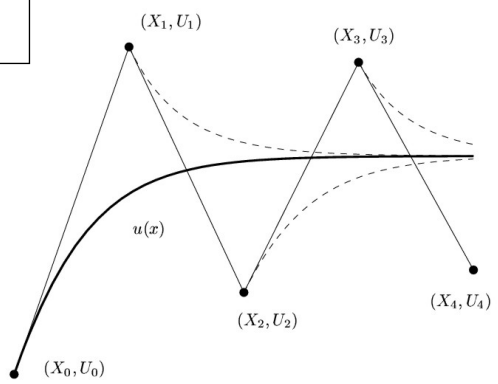
Euler explicite (Taylor n=1)
- Ordre de précision linéaire
- Mise en œuvre facile
- Stabilité ?

Taylor n quelconque
- Ordre de précision arbitrairement élevé,
- Mise en œuvre fastidieuse si n élevé
- Stabilité ?

18

Méthode de Taylor d'ordre 1
Euler explicite

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$



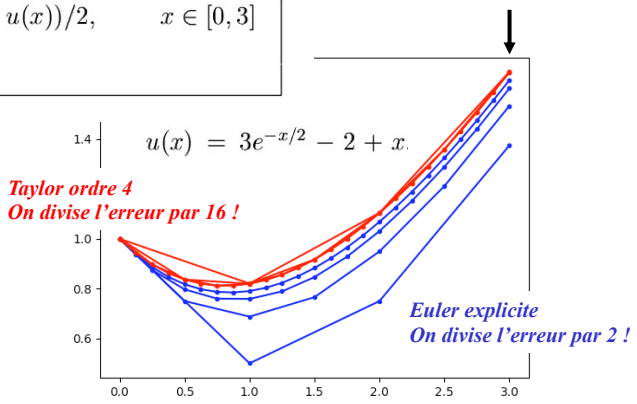
19

Exemple

Trouver $u(x)$ tel que

$$\begin{cases} u'(x) &= (x - u(x))/2, & x \in [0,3] \\ u(0) &= 1 \end{cases}$$

```
====== Explicit Euler method =====
==== Euler  (order=1) h=1.000 : eh(Xend) = 2.94e-01
==== Euler  (order=1) h=0.500 : eh(Xend) = 1.35e-01
==== Euler  (order=1) h=0.250 : eh(Xend) = 6.51e-02
==== Euler  (order=1) h=0.125 : eh(Xend) = 3.20e-02
============= Estimated order : 1.0678
====== Explicit Taylor order 4 method =====
==== Taylor (order=4) h=1.000 : eh(Xend) = 7.96e-04
==== Taylor (order=4) h=0.500 : eh(Xend) = 4.03e-05
==== Taylor (order=4) h=0.250 : eh(Xend) = 2.27e-06
==== Taylor (order=4) h=0.125 : eh(Xend) = 1.35e-07
============= Estimated order : 4.1767
```

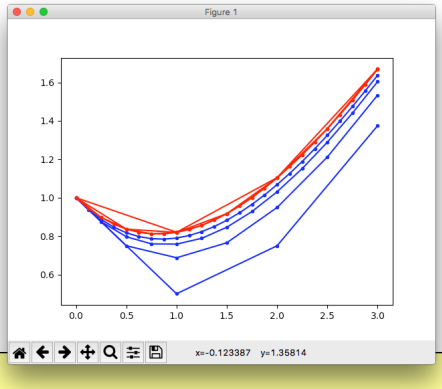


Taylor ordre 4
On divise l'erreur par 16 !

Euler explicite
On divise l'erreur par 2 !

20

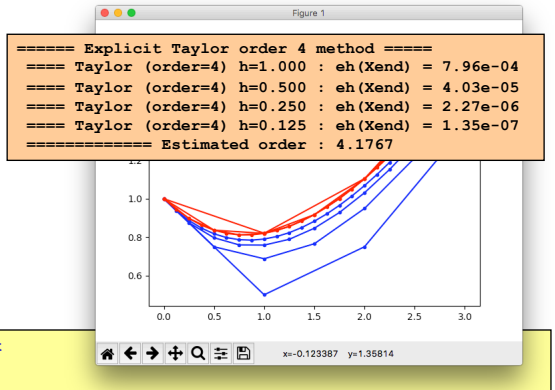
Taylor ordre 4 u'= (x-u)/2




```
u = lambda x : 3*exp(-x/2)-2+x

Xstart = 0; Xend = 3;
Ustart = 1;

for n in [3,6,12,24]:
  h = (Xend-Xstart)/n
  X = linspace(Xstart,Xend,n+1)
  U = zeros(n+1); U[0] = Ustart
  for i in range(n):
    U[i+1] =  U[i] + h*(X[i]-U[i])/2 + (h**2)*(2-X[i]+U[i])/8 \
        - (h**3)*(2-X[i]+U[i])/48 + (h**4)*(2-X[i]+U[i])/384
```

21

Calcul du taux de convergence



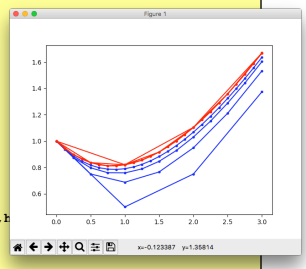

```
u = lambda x : 3*exp(-x/2)-2+x
error = zeros(4)
for j in range(4):
  n = 3*pow(2,j)
  h = (Xend-Xstart)/n
  X = linspace(Xstart,Xend,n+1)
  U = zeros(n+1); U[0] = 1
  for i in range(n):
    U[i+1] =  U[i] + h*(X[i]-U[i])/2 + (h**2)*(2-X[i]+U[i])/8 \
        - (h**3)*(2-X[i]+U[i])/48 + (h**4)*(2-X[i]+U[i])/384
  error[j] = abs(U[-1]-u(Xend))
  print(" ==== Taylor : h=%5.3f : eh(Xend) = %8.2e " % (h,error[j]))
order = mean(log(error[:-1]/error[1:])/log(2))
print(" ============ Estimated order : %.4f " % order)
```

22

Ne jamais dupliquer du code !

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

class ExplMethods(object):
  def __init__(self,name,color,f):
    self.name = name
    self.f = f
    self.color = color

integrators = [ExplMethods("Explicit Euler order 1",".-b",
                    lambda u,x,h : h*(x-u)/2  ),
               ExplMethods("Explicit Taylor order 4",".-r",
                    lambda u,x,h : h*(x-u)/2+(2-x+u)*((h**2)/8-(h**3)/48+(h**4)/384) ) ]

u = lambda x : 3*exp(-x/2)-2+x
Xstart = 0; Xend = 3; Ustart = 1;

for integrator in integrators:
  print(" ====== %s method=====" % integrator.name)
  error = zeros(4)
  for j in range(4):
    n = 3*pow(2,j)
    h = (Xend-Xstart)/n
    X = linspace(Xstart,Xend,n+1)
    U = zeros(n+1); U[0] = Ustart
    for i in range(n):
      U[i+1] =  U[i] + integrator.f(U[i],X[i],h)
    plt.plot(X,U,integrator.color)
    error[j] = abs(U[-1]-u(Xend))
    print(" ==== %s : h=%5.3f : eh(Xend) = %8.2e " % (integrator.name,h

  order = mean(log(error[:-1]/error[1:])/log(2))
  print(" ============= Estimated order : %.4f " % order)
```

Introduire une classe d'intégrateurs explicites



23

Comment estimer l'erreur ?

Le roi s'en va par une porte…

Développement en série de Taylor

$$u(X_{i+1}) = u(X_i) + h\, f(X_i, u(X_i)) + \frac{h^2}{2}\, u''(\xi_{i+1})$$

ξ_{i+1} est un point particulier de l'intervalle $[X_i, X_{i+1}]$

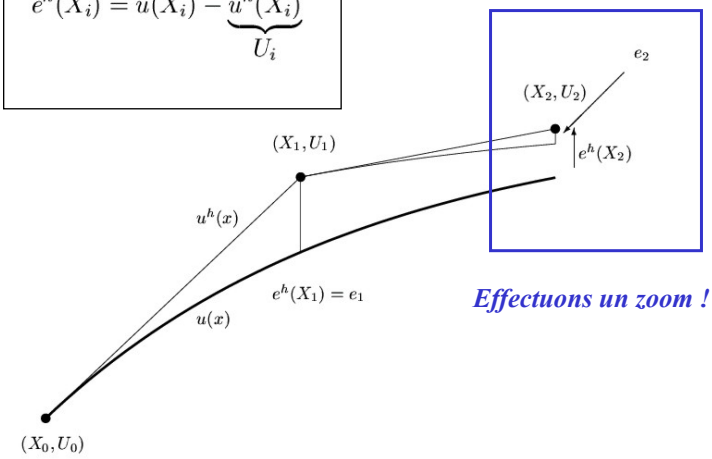
Et un triste substitut apparaît…

avec son cortège de courtisans…

24

Définissons l'erreur

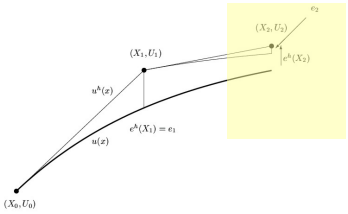
$$e^h(X_i) = u(X_i) - \underbrace{u^h(X_i)}_{U_i}$$



Effectuons un zoom !

25

Erreur locale et globale



Erreur locale commise lors du pas de X_1 à X_2

e_2

(X_2, U_2)

$e^h(X_2)$

Erreur globale de discrétisation = erreur propagée + erreur locale

26

Taylor
dit bonjour…

$$u(X_{i+1}) = u(X_i) + h\ f(X_i, u(X_i)) + \frac{h^2}{2}\ u''(\xi_{i+1})$$

$$\underbrace{u(X_{i+1}) - U_{i+1}}_{e^h(X_{i+1})} = \underbrace{u(X_i) - U_i}_{e^h(X_i)} + h\ \Big(f(X_i, u(X_i)) - f(X_i, U_i) \Big) + \frac{h^2}{2}\ u''(\xi_{i+1})$$

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$

… à Euler

27

$$\underbrace{u(X_{i+1}) - U_{i+1}}_{e^h(X_{i+1})} = \underbrace{u(X_i) - U_i}_{e^h(X_i)} + h\ \Big(f(X_i, u(X_i)) - f(X_i, U_i) \Big) + \frac{h^2}{2}\ u''(\xi_{i+1})$$

En vertu du théorème de la valeur moyenne,
il existe ζ_i dans l'intervalle entre U_i et $u(X_i)$
si f est continue et différentiable par rapport à u.

$$e^h(X_{i+1}) = e^h(X_i) + h\ \underbrace{\Big(u(X_i) - U_i \Big)}_{e^h(X_i)} \left.\frac{\partial f}{\partial u}\right|_{(X_i, \zeta_i)} + \frac{h^2}{2}\ u''(\xi_{i+1})$$

En définissant $J_i = \left.\frac{\partial f}{\partial u}\right|_{(X_i, \zeta_i)}$ et $e_{i+1} = \frac{h^2}{2}\ u''(\xi_{i+1})$,

$$e^h(X_{i+1}) = e^h(X_i)\ \Big(1 + hJ_i \Big) + e_{i+1}$$

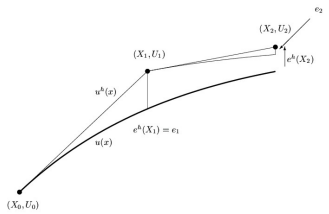
Erreur locale

Erreur propagée

28

$$a_i = (1 + hJ_i)$$

Facteur d'amplification ou d'amortissement (cas du dessin !) des erreurs précédentes



Propagation des erreurs…

$$\begin{aligned} e^h(X_1) &= e_1 \\ e^h(X_2) &= a_1\, e_1 + e_2 \\ e^h(X_3) &= a_2 a_1\, e_1 + a_2\, e_2 + e_3 \end{aligned}$$

$$\dots$$

$$e^h(X_{i+1}) = \underbrace{a_i a_{i-1} \dots a_2 a_1\, e_1 + a_i \dots a_2\, e_2 \quad \dots + a_i e_i}_{\text{erreur propagée}} + e_{i+1}$$

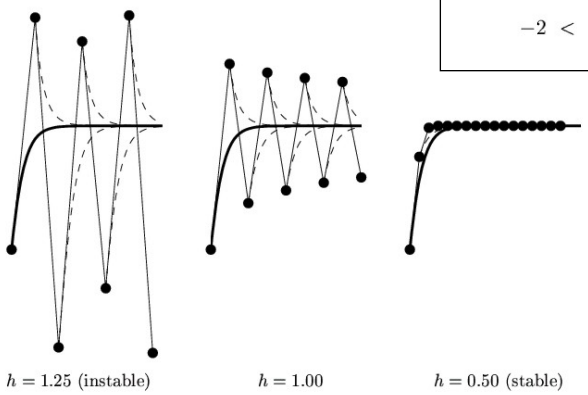
29

Stabilité de la méthode d'Euler

$$\overbrace{|1 + hJ_i|}^{a_i} < 1 \qquad \forall i,$$

$$\Downarrow$$

$$-2 < hJ_i < 0 \qquad \forall i,$$



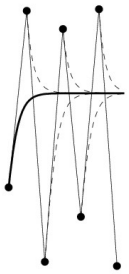
$h = 1.25$ (instable) $h = 1.00$ $h = 0.50$ (stable)

30

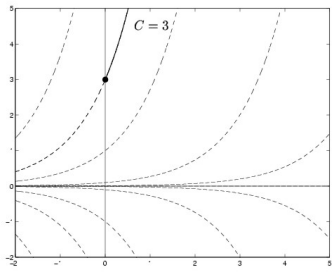
Stabilité…

Stabilité d'un système physique :
systèmes chaotiques, turbulence…

Stabilité d'une méthode numérique :
instabilité numérique de la méthode d'Euler explicite



Stabilité d'un modèle mathématique :
sensibilité aux données



Modélisation mathématique

Simulation numérique

31

Plan du cours de méthodes numériques

Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non linéaires ?

Et les méthodes itératives ?

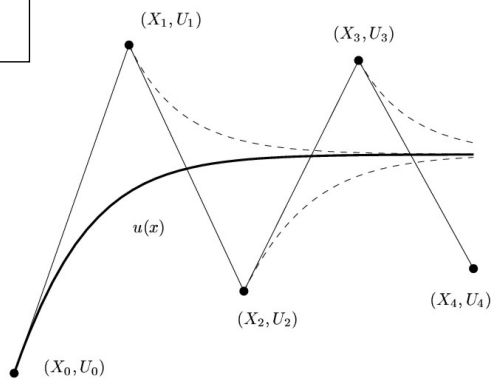
Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Méthode d'Euler explicite

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$



Cas général dans le plan complexe

Nombre complexe

$$U_{i+1} = \left(1 + h\lambda\right) U_i$$

$$\begin{cases} u' = \lambda u \\ u(0) = 1 \end{cases}$$

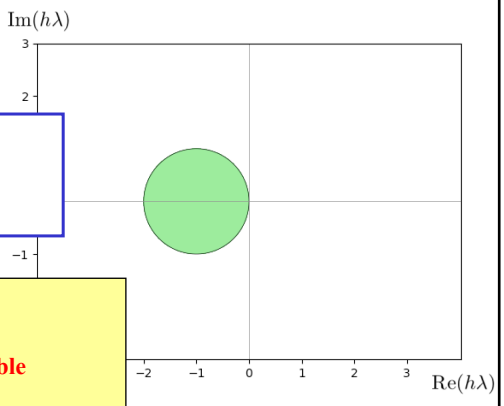
Problème modèle correspondant à la linéarisation en un instant

Euler explicite
Ordre de précision linéaire
Méthode conditionnellement stable

Région de stabilité Euler explicite

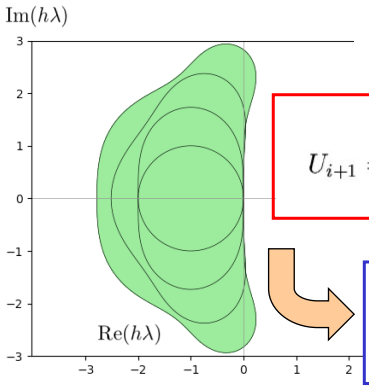
$$U_{i+1} = \left(1 + h\lambda\right) U_i$$

$$|1 + h\lambda| < 1$$



Euler explicite
Ordre de précision linéaire
Méthode conditionnellement stable

Région de stabilité Méthodes de Taylor



$$U_{i+1} = \left(1 + h\lambda + \frac{h^2\lambda^2}{2!} + \cdots + \frac{h^n\lambda^n}{n!}\right) U_i$$

$$\left|1 + h\lambda + \frac{h^2\lambda^2}{2!} + \cdots + \frac{h^n\lambda^n}{n!}\right| < 1$$

Taylor n quelconque

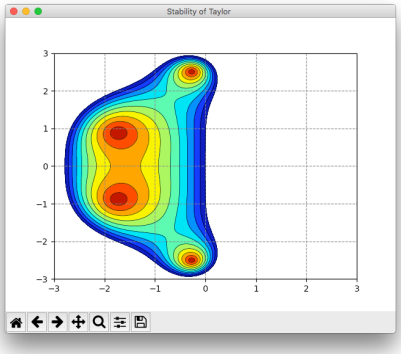
- **Ordre de précision arbitrairement élevé,**
- **Mise en œuvre fastidieuse si n élevé**
- **Conditionnellement stables**

Comment obtenir un tel graphe ?

Les couleurs ont du sens !

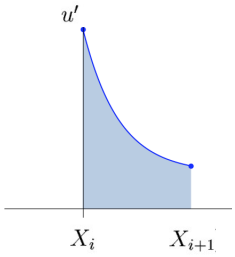
- **Bleu : pas d'amplification !**
- **Rouge : amortissement !**

Cela concernera l'erreur mais aussi la solution exacte.
Une méthode numérique TROP stable n'a aucun intérêt, puisqu'elle gomme les erreurs mais aussi la solution !
C'est à la frontière que le comportement est optimal.




```
x,y = meshgrid(linspace(-3,3,1000),linspace(-3,3,1000))
z = x + 1j*y

gain = abs(1 + z + z*z/2 + z**3/6 + z**4/24)
plt.contourf(x,y,gain,arange(0,1.1,0.1),cmap=plt.cm.jet_r)
```

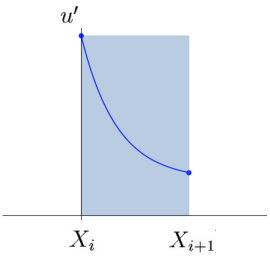
Euler explicite revisitée…



$$\int_{X_i}^{X_{i+1}} f(x,u(x))dx = u(X_{i+1}) - u(X_i)$$

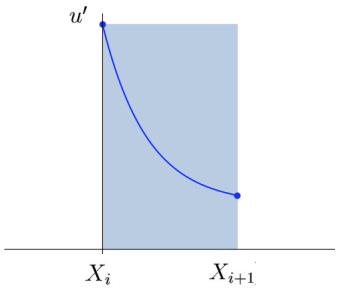
En approximant l'intégrale par un rectangle,

$$h\, f(X_i, U_i) \approx U_{i+1} - U_i$$



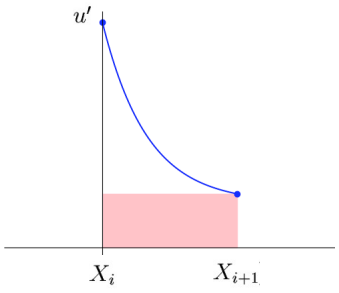
Idée

Il existe d'autres manières d'estimer cette intégrale !



Euler implicite

$$U_{i+1} - U_i = h\lambda\, U_{i+1}$$



$$U_{i+1} - U_i = h\lambda\, U_i$$

Euler explicite

$$U_{i+1}\left(1-h\lambda\right)=U_i$$

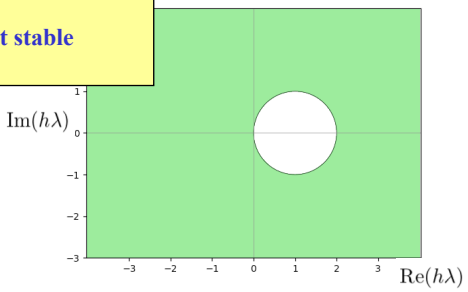
Région de stabilité Euler implicite

$$\left|\frac{1}{1-h\lambda}\right| < 1$$

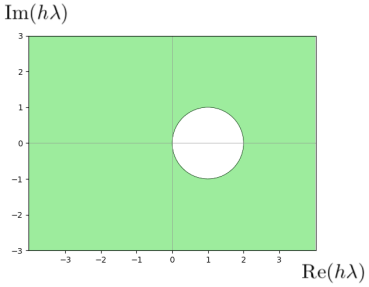
Euler implicite

Ordre de précision linéaire

Méthode inconditionnellement stable



La douce illusion de l'inconditionnellement stable...



Exemple :

u'(x) = sin(u(x))

$$U_{i+1} = U_i + h \sin U_{i+1}$$

Difficile, difficile, très difficile !

Euler implicite

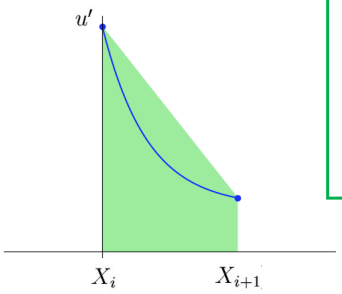
Ordre de précision linéaire

Equation à résoudre à chaque pas de temps

(équation non linéaire, si f non linéaire !)

Inconditionnellement stable

Méthode de Crank-Nicolson



$$U_{i+1} - U_i = h\lambda \left(\frac{U_i + U_{i+1}}{2} \right)$$

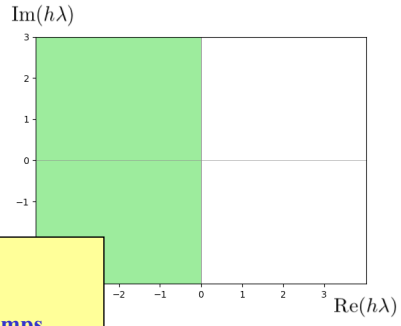
Crank-Nicolson (trapèzes)
- Ordre de précision quadratique
- Equation à résoudre à chaque pas de temps
  (équation non linéaire, si f non linéaire !)
- Inconditionnellement stable

Stabilité et précision

$$U_{i+1}\left(1 - h\lambda/2\right) = U_i\left(1 + h\lambda/2\right)$$

$$\left|\frac{1 + h\lambda/2}{1 - h\lambda/2}\right| < 1$$

$$\Re(h\lambda) < 0$$



Crank-Nicolson (trapèzes)
- Ordre de précision quadratique
- Equation à résoudre à chaque pas de temps
  (équation non linéaire, si f non linéaire !)
- Inconditionnellement stable

Leapfrog Method

$$U_{i+1} = U_{i-1} + 2hf(X_i, U_i)$$

Méthode du Saute-Mouton
- **Méthode à pas liés**
- **Comment démarrer ?**
- **Ordre de précision ?**
- **Stabilité ?**

$$u'(X_i) \approx \frac{U_{i+1} - U_{i-1}}{2h}$$

Il s'agit d'une méthode à pas liés car pour calculer U_{i+1}, il est nécessaire de connaître U_i et U_{i-1}.

Comment démarrer ?

Cette méthode peut être utilisée seulement lorsqu'on possède U_0 et U_1.
Il n'est donc pas possible de commencer directement le calcul.

1 pas d'une méthode à pas simple
- **Euler explicite ou implicite**
- **Crank-Nicolson**
- **Méthodes de Taylor d'ordre élevé**
- **Méthodes de Gear : on va y arriver !**

Et puis autant de pas de la méthode à pas double
- **Leapfrog**
- **Méthodes de Adams-Bashfort-Moulton : on va y arriver !**

Précision ?

$$u'(X_i) = f(X_i, U_i)$$

En approximant $u'(X_i)$ par la différence centrée d'ordre deux

$$\frac{U_{i+1} - U_{i-1}}{2h} + \mathcal{O}(h^2) = f(X_i, U_i)$$

$$U_{i+1} = U_{i-1} + 2hf(X_i, U_i) + \mathcal{O}(h^3)$$

Méthode d'ordre deux

On gagne un ordre de précision en utilisant une différence centrée pour estimer la dérivée, contrairement aux méthodes d'Euler.

Les trapèzes sont une intégration centrée !
Leapfrog est une dérivation centrée !

Stabilité ?

a = facteur d'amplification de l'erreur initiale à chaque itération

$$U_i = a^i U_0$$

Changement de variable pour l'analyse des schémas à pas liés

$$\begin{cases} u' = \lambda u \\ u(0) = U_0 \end{cases}$$

Problème modèle correspondant à la linéarisation en un instant

$$2h\lambda a^i U_0 = a^{i+1} U_0 - a^{i-1} U_0$$

$$
\begin{aligned}
2h\lambda a^i U_0 &= a^{i+1}U_0 - a^{i-1}U_0 \\
&\downarrow \quad \textit{En simplifiant par } a^{i-1}U_0 \ , \\
a^2 - 2h\lambda a - 1 &= 0 \\
a &= h\lambda \pm \sqrt{h^2\lambda^2 + 1}
\end{aligned}
$$

Région stable

$$|h\lambda \pm \sqrt{h^2\lambda^2 + 1}| < 1$$

Egalité obtenue pour le segment de –i à i
Dans cette zone, pas d'accroissement ou d'amortissement des perturbations…

Systèmes d'équations différentielles

$$
\begin{cases}
u_1'(x) &= f_1(x, u_1(x), u_2(x), \ldots, u_n(x)) \\
u_2'(x) &= f_2(x, u_1(x), u_2(x), \ldots, u_n(x)) \\
\quad\vdots & \\
u_n'(x) &= f_n(x, u_1(x), u_2(x), \ldots, u_n(x))
\end{cases}
$$

$$
\begin{cases}
u_1(a) &= \overline{u}_1 \\
u_2(a) &= \overline{u}_2 \\
\quad\vdots & \\
u_n(a) &= \overline{u}_n
\end{cases}
$$

Notation compacte : les vecteurs sont en gras.

Trouver $\mathbf{u}(x)$ tel que

$$
\begin{cases}
\mathbf{u}'(x) &= \mathbf{f}(x, \mathbf{u}(x)), \qquad x \in [a, b] \\
\mathbf{u}(a) &= \overline{\mathbf{u}}
\end{cases}
$$

C'est exactement la même histoire !

$$U_{ji} = u_j^h(X_i) \approx u_j(X_i)$$

$$\mathbf{U}_i = \mathbf{u}^h(X_i) \approx \mathbf{u}(X_i)$$

Notation compacte : les vecteurs sont en gras.

Méthode d'Euler explicite

$$U_{ji+1} = U_{ji} + hf_j(X_i, U_{1i}, \ldots, U_{ni})$$

Tableau à deux indices
On calcule la j-ème composante du vecteur des inconnues à la i-ème abscisse temporelle

Vecteur
On calcule la j-ème composante du vecteur des inconnues.

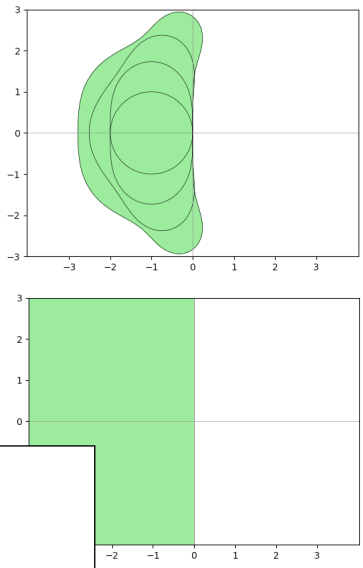
Stabilité des systèmes

$$\mathbf{u}' = \mathbf{A}\mathbf{u}$$

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$$

$$\mathbf{v}(x) = \mathbf{P}^{-1}\mathbf{u}(x)$$

$$\mathbf{v}' = \mathbf{D}\mathbf{v}$$



Le problème différentiel initial est équivalent à n équations scalaires

$$v_i'(x) = \lambda_i v_i(x), \qquad i = 1, \ldots, n$$

Trouver $u(x)$ tel que

$$\begin{cases} u'(x) = f(x, u(x)), & x \in [a, b] \\ u(a) = \bar{u} \end{cases}$$

Problème de Cauchy…

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + \tfrac{h}{2}, U_i + \tfrac{h}{2}K_1) \\ K_3 &= f(X_i + \tfrac{h}{2}, U_i + \tfrac{h}{2}K_2) \\ K_4 &= f(X_i + h, U_i + hK_3) \end{aligned}$$

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{2}(K_1 + K_2) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + h, \underbrace{U_i + hK_1}_{P_2}) \end{aligned}$$

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$

… et les méthodes de Runge-Kutta (ode45)

Méthodes explicites de Runge-Kutta

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{2}(K_1 + K_2) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + h, \underbrace{U_i + hK_1}_{P_2}) \end{aligned}$$

Heun (Runge-Kutta n=2)

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + \tfrac{h}{2}, U_i + \tfrac{h}{2}K_1) \\ K_3 &= f(X_i + \tfrac{h}{2}, U_i + \tfrac{h}{2}K_2) \\ K_4 &= f(X_i + h, U_i + hK_3) \end{aligned}$$

"le classique" (Runge-Kutta n=4)

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$

Euler explicite (Runge-Kutta n=1)

Runge-Kutta n quelconque
- **Ordre de précision arbitrairement élevé,**
- **Facile à mettre en oeuvre**
- **Conditionnellement stables (comme Taylor)**

Méthode de Heun
Comment choisir les paramètres ?

$$
\begin{aligned}
U_{i+1} &= U_i + h(w_1 K_1 + w_2 K_2) \\
K_1 &= f(X_i, U_i) \\
K_2 &= f(X_i + \alpha h, U_i + \beta h K_1)
\end{aligned}
$$

$$
\begin{aligned}
U_{i+1} &= U_i + \tfrac{h}{2}(K_1 + K_2) \\
K_1 &= f(X_i, U_i) \\
K_2 &= f(X_i + h, \underbrace{U_i + hK_1}_{P_2})
\end{aligned}
$$

Il faut la précision requise…

$$
\begin{aligned}
U_{i+1} &= U_i + h\, f(X_i, U_i) \\
&\quad + \frac{h^2}{2}\left(\frac{\partial f}{\partial x}(X_i, U_i) + \frac{\partial f}{\partial u}(X_i, U_i)\, f(X_i, U_i)\right) + \mathcal{O}(h^3)
\end{aligned}
$$

$$
\begin{aligned}
U_{i+1} &= U_i + h(w_1 K_1 + w_2 K_2) \\
U_{i+1} &= U_i + h w_1 f(X_i, U_i) + h w_2 f(X_i + \alpha h, U_i + \beta h K_1)
\end{aligned}
$$

En effectuant un développement en série de Taylor de l'expression $f(X_i + \alpha h, U_i + \beta h K_1)$, autour de (X_i, U_i),

$$
\begin{aligned}
U_{i+1} &= U_i + h w_1\, f(X_i, U_i) \\
&\quad + h w_2 \left(f(X_i, U_i) + \alpha h \frac{\partial f}{\partial x}(X_i, U_i) + \beta h \frac{\partial f}{\partial u}(X_i, U_i)\, f(X_i, U_i) + \mathcal{O}(h^2)\right)
\end{aligned}
$$

$$
\begin{aligned}
U_{i+1} &= U_i + h(w_1 + w_2)\, f(X_i, U_i) \\
&\quad + h^2\, w_2 \left(\alpha \frac{\partial f}{\partial x}(X_i, U_i) + \beta \frac{\partial f}{\partial u}(X_i, U_i)\, f(X_i, U_i)\right) + \mathcal{O}(h^3)
\end{aligned}
$$

Identification des termes
3 relations à satisfaire
4 paramètres à choisir

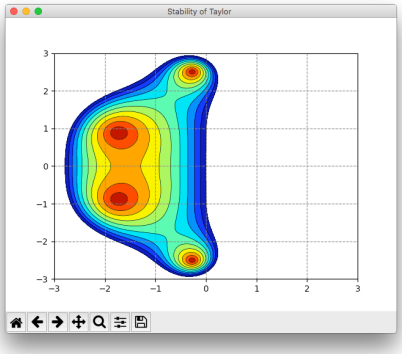
Il existe plusieurs possibilités !

Les zones de stabilité de Runge-Kutta et de Taylor sont identiques...

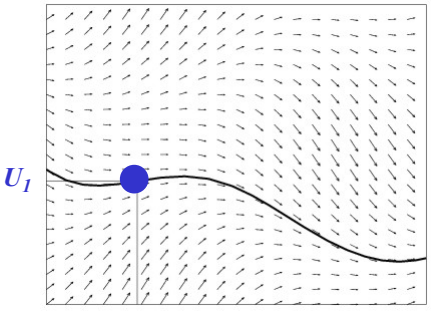
On a construit les méthodes de Runge-Kutta en identifiant les développements en série de Taylor... En conséquence, les méthodes de Runge-Kutta et de Taylor sont IDENTIQUES pour un problème linéaire !

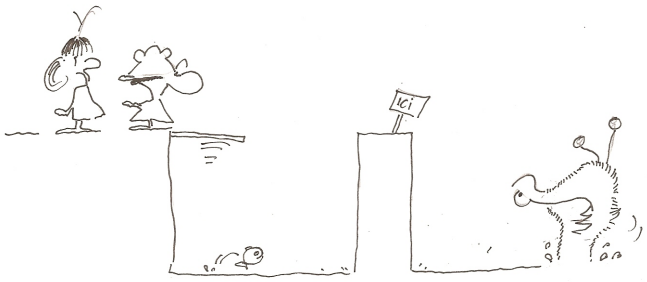
Mais, elles ne sont PAS identiques pour des problèmes non linéaires.

Finalement, comme l'analyse de stabilité se fait sur un problème linéaire, cette analyse fournira des résultats IDENTIQUES... Vérifiez ce résultat par vous-même en reproduisant le calcul effectué pour la méthode Leapfrog :-)

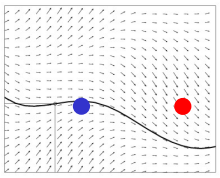


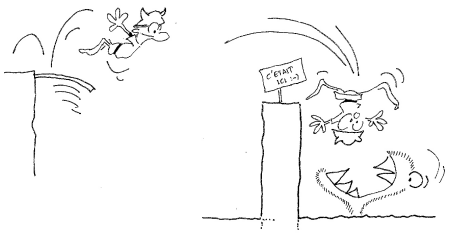
Interprétation des méthodes de Runge-Kutta

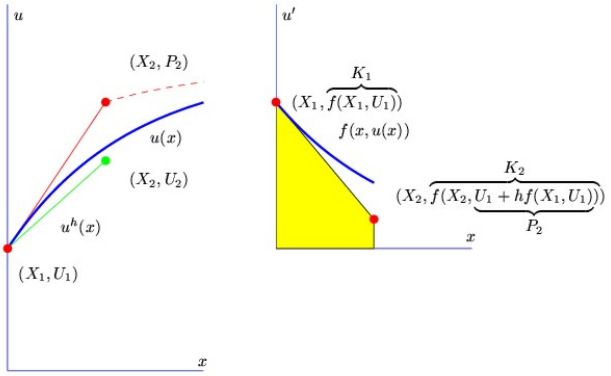




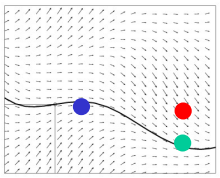
Prédiction

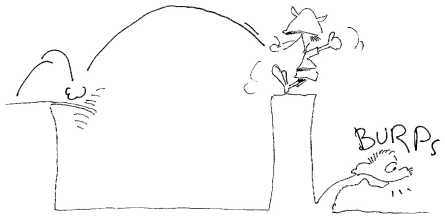


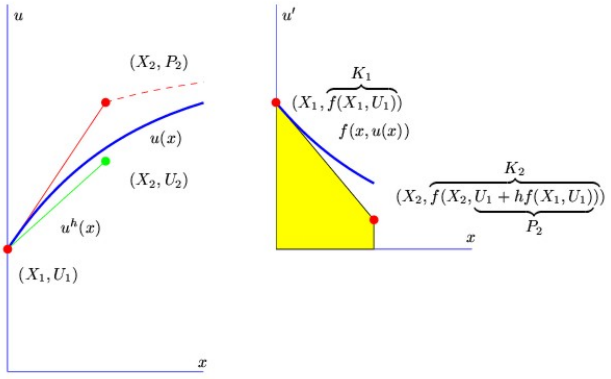




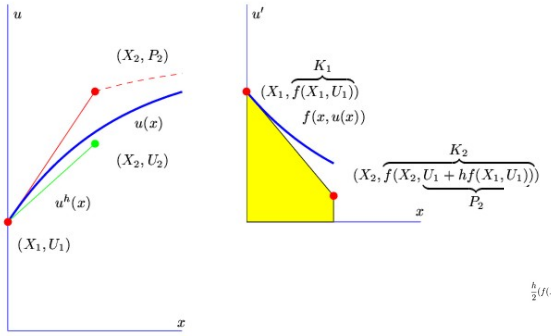
Correction







Prédiction

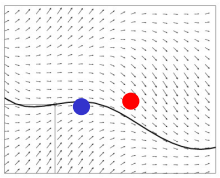


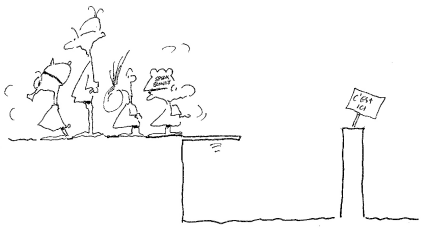
$$
\begin{aligned}
f(X_1, U_1) &= u'(X_1) \\
&\downarrow \text{ En utilisant une différence décentrée pour estimer } u'(X_1), \\
f(X_1, U_1) &\approx \frac{U(X_2) - U_1}{h} \\
&\downarrow \text{ En approximant } u(X_2) \text{ par } P_2, \\
f(X_1, U_1) &\approx \frac{P_2 - U_1}{h} \\
&\downarrow \\
U_1 + h\underbrace{f(X_1, U_1)}_{K_1} &\approx P_2
\end{aligned}
$$

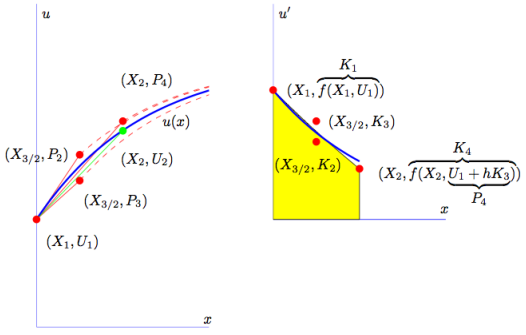
$$
\begin{aligned}
\int_{X_1}^{X_2} f(x, u(x))dx &= u(X_2) - u(X_1) \\
&\downarrow \text{ En utilisant la méthode des trapèzes,} \\
\frac{h}{2}(f(X_1, u(X_1)) + f(X_2, u(X_2))) &\approx u(X_2) - u(X_1) \\
&\downarrow \text{ En approximant } u(X_2) \text{ par } P_2 \text{ pour l'intégrale,} \\
&\quad \text{et par } U_2 \text{ à droite,} \\
\frac{h}{2}(\underbrace{f(X_1, U_1)}_{K_1} + \underbrace{f(X_2, P_2)}_{K_2}) &\approx U_2 - U_1 \\
&\downarrow \\
U_1 + \frac{h}{2}(K_1 + K_2) &\approx U_2
\end{aligned}
$$

Correction

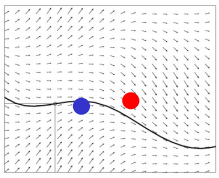
Prédiction

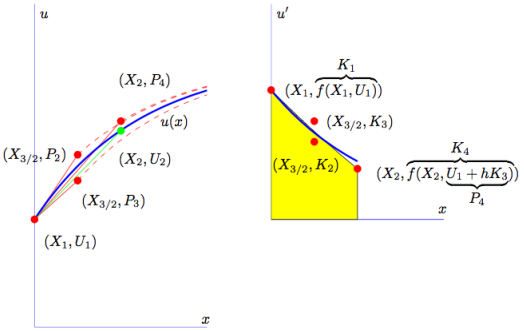




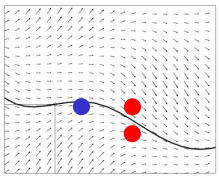


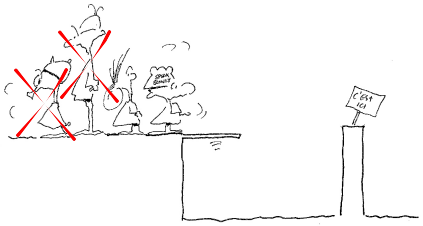
Prédiction

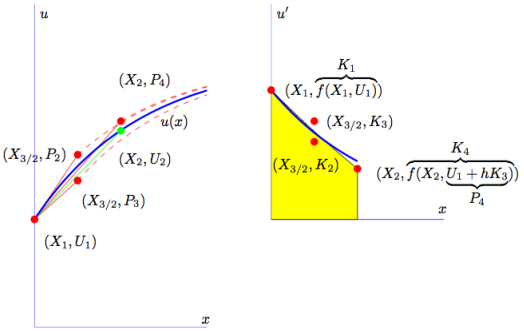




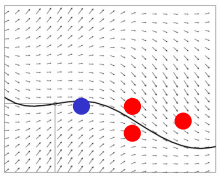
Prédiction

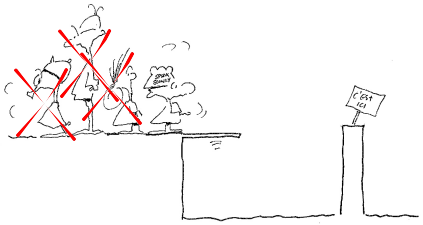


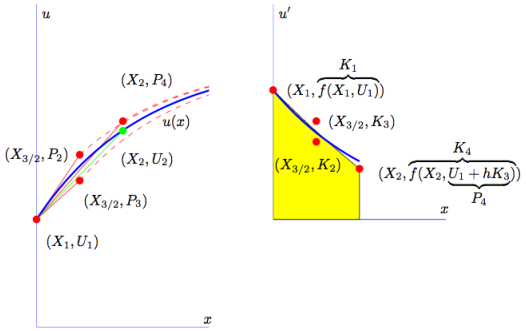




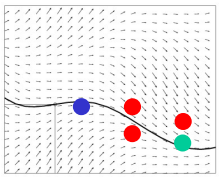
Prédiction

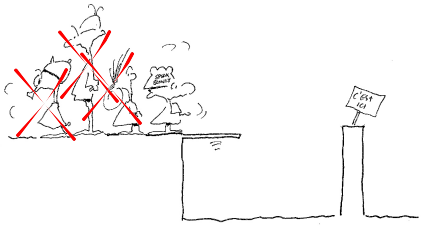


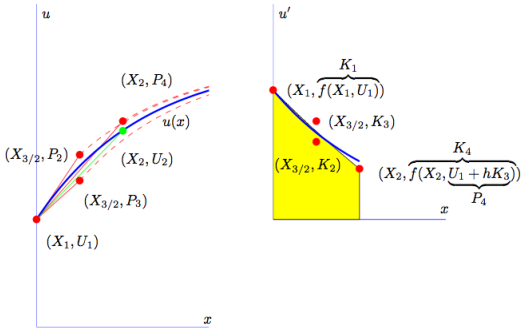




Correction







Runge-Kutta-Fehlberg

$$
\begin{aligned}
P_{i+1} &= U_i + h(\tfrac{25}{216}K_1 + \tfrac{1408}{2565}K_3 + \tfrac{2197}{4104}K_4 - \tfrac{1}{5}K_5) \\
U_{i+1} &= U_i + h(\tfrac{16}{235}K_1 + \tfrac{6656}{12825}K_3 + \tfrac{28561}{56430}K_4 - \tfrac{9}{50}K_5 + \tfrac{2}{55}K_6) \\
K_1 &= f(X_i, U_i) \\
K_2 &= f(X_i + \tfrac{h}{4}, U_i + \tfrac{h}{4}K_1) \\
K_3 &= f(X_i + \tfrac{3h}{8}, U_i + \tfrac{3h}{32}K_1 + \tfrac{9h}{32}K_2) \\
K_4 &= f(X_i + \tfrac{12h}{13}, U_i + \tfrac{1932h}{2197}K_1 - \tfrac{7200h}{2197}K_2 + \tfrac{7296h}{2197}K_3) \\
K_5 &= f(X_i + h, U_i + \tfrac{439h}{216}K_1 - 8hK_2 + \tfrac{3680h}{513}K_3 - \tfrac{845h}{4104}K_4) \\
K_6 &= f(X_i + \tfrac{h}{2}, U_i - \tfrac{8h}{27}K_1 + 2hK_2 - \tfrac{3544h}{2565}K_3 - \tfrac{1859h}{4104}K_4 - \tfrac{11h}{40}K_5)
\end{aligned}
$$

Calcul simultané et efficace de deux approximations de la valeur

P_{i+1} une approximation d'ordre 4
U_{i+1} une approximation d'ordre 5

La différence permet d'obtenir une estimation de l'erreur locale commise

Problème de Cauchy

Trouver $u(x)$ tel que

$$
\begin{cases} u'(x) = f(x, u(x)), & x \in [a, b] \\ u(a) = \bar{u} \end{cases}
$$

Méthodes explicites

Rien à résoudre (cool !)
Stabilité conditionnelle (embêtant !)

Méthodes implicites

Il faut résoudre un problème (dur, si non-linéaire !)
Stables (cool !) (parfois trop !)

Méthodes explicites de Taylor

Il faut disposer de nombreuses dérivées de f !
Stabilité conditionnelle

Méthodes explicites de Runge-Kutta

Il faut seulement disposer de la fonction f
Stabilité (théoriquement, idem que Taylor)

Valeur exacte

$$u(X_i) \approx u^h(X_i) = U_i$$

Approximation numérique

Méthodes explicites de Runge-Kutta

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{2}(K_1 + K_2) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + h, \underbrace{U_i + hK_1}_{P_2}) \end{aligned}$$

Heun (Runge-Kutta n=2)

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + \tfrac{h}{2}, U_i + \tfrac{h}{2}K_1) \\ K_3 &= f(X_i + \tfrac{h}{2}, U_i + \tfrac{h}{2}K_2) \\ K_4 &= f(X_i + h, U_i + hK_3) \end{aligned}$$

"le classique" (Runge-Kutta n=4)

$$U_{i+1} = U_i + h\ f(X_i, U_i)$$

Euler explicite (Runge-Kutta n=1)

Runge-Kutta n quelconque

- **Ordre de précision arbitrairement élevé,**
- **Facile à mettre en oeuvre**
- **Conditionnellement stables (comme Taylor)**

Runge-Kutta-Fehlberg

$$\begin{aligned} P_{i+1} &= U_i + h(\tfrac{25}{216}K_1 + \tfrac{1408}{2565}K_3 + \tfrac{2197}{4104}K_4 - \tfrac{1}{5}K_5) \\ U_{i+1} &= U_i + h(\tfrac{16}{235}K_1 + \tfrac{6656}{12825}K_3 + \tfrac{28561}{56430}K_4 - \tfrac{9}{50}K_5 + \tfrac{2}{55}K_6) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + \tfrac{h}{4}, U_i + \tfrac{h}{4}K_1) \\ K_3 &= f(X_i + \tfrac{3h}{8}, U_i + \tfrac{3h}{32}K_1 + \tfrac{9h}{32}K_2) \\ K_4 &= f(X_i + \tfrac{12h}{13}, U_i + \tfrac{1932h}{2197}K_1 - \tfrac{7200h}{2197}K_2 + \tfrac{7296h}{2197}K_3) \\ K_5 &= f(X_i + h, U_i + \tfrac{439h}{216}K_1 - 8hK_2 + \tfrac{3680h}{513}K_3 - \tfrac{845h}{4104}K_4) \\ K_6 &= f(X_i + \tfrac{h}{2}, U_i - \tfrac{8h}{27}K_1 + 2hK_2 - \tfrac{3544h}{2565}K_3 - \tfrac{1859h}{4104}K_4 - \tfrac{11h}{40}K_5) \end{aligned}$$

Calcul simultané et efficace de deux approximations de la valeur

P_{i+1} une approximation d'ordre 4
U_{i+1} une approximation d'ordre 5

La différence permet d'obtenir une estimation de l'erreur locale commise

Comment choisir un nouveau pas h_{i+1} ?

On construit une stratégie adaptative basée sur des arguments heuristiques peu rigoureux...

Stratégie prudente ou intrépide selon votre caractère

Estimation de l'erreur locale que l'on vient de commettre au pas précédent h_i

Relations heuristiques

$$\theta_i = Ch_i^{n+1}$$

$$\theta_{i+1} = Ch_{i+1}^{n+1}$$

$$\theta_{i+1} \approx \theta_i \left(\frac{h_{i+1}}{h_i}\right)^{n+1}$$

$$h_{i+1} \leq h_i \left(\frac{\epsilon}{\theta_i}\right)^{\frac{1}{(n+1)}}$$

Tolérance
Erreur acceptable

$$\theta_{i+1} \leq \epsilon$$

Estimation de l'erreur que l´on commettra au pas suivant : elle doit satisfaire le critère

Stratégie adaptative

1. Estimation de l'erreur commise à l'étape courante i avec le pas h_i en effectuant le calcul de θ_i. On utilise la différence entre la prédiction et la correction pour effectuer cette estimation.

$$\theta_i = \left| U_{i+1} - P_{i+1} \right|$$

2. Est-ce que le critère d'erreur est respecté ? Si $\theta_i > \epsilon$, il faut recommencer le pas courant i avec un plus petit pas de temps. On choisira comme nouveau pas $h = h_i/2$. Il convient aussi de vérifier que le nouveau pas de temps n'est pas inférieur à une limite minimale h_{min} fixée a priori. Si cela arrive, le programme s'arrête de manière définitive avec un constat d'échec !

3. Estimation d'un nouveau pas optimal $\widehat{h}$ à partir de l'estimation θ_i de l'erreur locale qui vient d'être commise au pas précédent.

$$\widehat{h} = h_i \left(\frac{\epsilon}{\theta_i}\right)^{\frac{1}{(n+1)}}$$

4. Si $\widehat{h} > h_i$, on choisit comme nouveau pas : $h_{i+1} = min(\widehat{h}, \frac{3}{2}h_i, h_{max})$

5. Si $\widehat{h} < h_i$, on choisit comme nouveau pas : $h_{i+1} = max(\widehat{h}, \frac{1}{2}h_i, h_{min})$

Méthodes d'Adams

Idée des méthodes à pas liés

On souhaite exploiter la connaissance des valeurs passées.

$$U_{i+1} = U_i + h \sum_{j=-1}^{n} \beta_j \underbrace{f(X_{i-j}, U_{i-j})}_{F_{i-j}} \qquad (i \geq n).$$

On utilise n+1 valeurs précédentes : Méthodes à pas liés

Idée 1 des méthodes prédicteurs-correcteurs

Obtenir une estimation de l'erreur locale commise

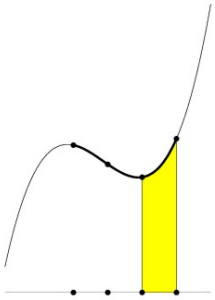
Idée 2 des méthodes prédicteurs-correcteurs

Obtenir une estimation de U_{i+1} pour les formules des méthodes implicites

Adams-Bashfort-Moulton n quelconque

- **Ordre de précision arbitrairement élevé,**
- **Prédicteur explicite à pas liés,**
- **Correcteur (semi-)implicite à pas liés,**
- **Conditionnellement stables**

Interpolation

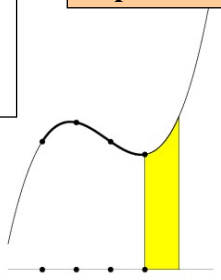


Adams-Moulton $\mathcal{O}(h^4)$

Implicite

$$U_{i+1} = U_i + \int_{X_i}^{X_{i+1}} f(x, u(x))dx$$

Explicite



Adams-Bashforth $\mathcal{O}(h^4)$

Extrapolation

Pratiquement...

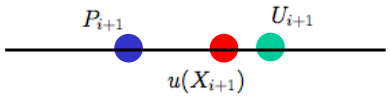
$$
\begin{aligned}
U_{i+1} &= U_i + h\ F_i \\
|e_i| &\leq \tfrac{C_2}{2}\ h^2 \qquad \text{(Euler explicite - Adams-Bashforth d'ordre 1)} \quad \mathcal{O}(h) \\
U_{i+1} &= U_i + h\ F_{i+1} \\
|e_i| &\leq \tfrac{C_2}{2}\ h^2 \qquad \text{(Euler implicite - Adams-Moulton d'ordre 1)} \quad \mathcal{O}(h)
\end{aligned}
$$

$$
\begin{aligned}
U_{i+1} &= U_i + \frac{h}{2}\left(-F_{i-1} + 3F_i\right) \\
|e_i| &\leq \tfrac{5C_3}{12}\ h^3 \qquad \text{(Adams-Bashforth d'ordre 2)} \quad \mathcal{O}(h^2) \\
U_{i+1} &= U_i + \frac{h}{2}\left(F_i + F_{i+1}\right) \\
|e_i| &\leq \tfrac{C_3}{12}\ h^3 \qquad \text{(trapèzes - Adams-Moulton d'ordre 2)} \quad \mathcal{O}(h^2)
\end{aligned}
$$

$$P_{i+1} = U_i + \frac{h}{24}\left(-9f(X_{i-3}, U_{i-3}) + 37f(X_{i-2}, U_{i-2}) - 59f(X_{i-1}, U_{i-1}) + 55f(X_i, U_i)\right)$$

$$U_{i+1} = U_i + \frac{h}{24}\left(f(X_{i-2}, U_{i-2}) - 5f(X_{i-1}, U_{i-1}) + 19f(X_i, U_i) + 9f(X_{i+1}, P_{i+1})\right)$$

Non, non : il n'y a pas d'erreur dans l'estimation d'erreur !



$$
\begin{aligned}
u(X_{i+1}) - P_{i+1} &= \frac{251h^5}{720}u^{(5)}(\xi_{i+1}), \\
u(X_{i+1}) - U_{i+1} &= \frac{-19h^5}{720}u^{(5)}(\zeta_{i+1}).
\end{aligned}
$$

$$(u(X_{i+1}) - U_{i+1})^2 \approx \underbrace{\left(\frac{-19}{270}\left(U_{i+1} - P_{i+1}\right)\right)^2}_{\theta_{i+1}^2}$$

Méthodes de Gear

$$u_{i+1}^h(x) = \sum_{j=0}^{n} U_{i+1-j}\ \phi_j(x)$$

$$u'(X_{i+1}) = \frac{f(X_{i+1}, \overbrace{u(X_{i+1})}^{U_{i+1}})}{F_{i+1}}$$

$$\downarrow$$

$$(u_{i+1}^h)'(X_{i+1}) \approx F_{i+1}$$

$$\downarrow$$

$$\sum_{j=0}^{n} U_{i+1-j}\ \phi_j'(X_{i+1}) \approx F_{i+1}$$

$$\downarrow$$

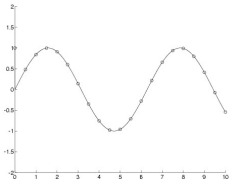
$$U_{i+1} \approx \frac{1}{\phi_0'(X_{i+1})}\left(-\sum_{j=1}^{n} U_{i+1-j}\ \phi_j'(X_{i+1}) + F_{i+1}\right)$$

Gear n quelconque

- **Ordre de précision arbitrairement élevé,**
- **Implicites à pas liés**
- **Très stables (ok pour problèmes raides)**

Exemple

$$\begin{aligned}
u_{i+1}^h(x) &= U_{i+1}\,\phi_0(x) + U_i\,\phi_1(x) + U_{i-1}\,\phi_2(x) \\
&= U_{i+1}\,\frac{(x-X_i)(x-X_{i-1})}{2h^2} \\
&\quad + U_i\,\frac{(x-X_{i+1})(x-X_{i-1})}{-h^2} + U_{i-1}\,\frac{(x-X_{i+1})(x-X_i)}{2h^2} \\
&\downarrow \\
(u_{i+1}^h)'(x) &= U_{i+1}\,\frac{(2x-X_i-X_{i-1})}{2h^2} \\
&\quad + U_i\,\frac{(2x-X_{i+1}-X_{i-1})}{-h^2} + U_{i-1}\,\frac{(2x-X_{i+1}-X_i)}{2h^2} \\
&\downarrow \\
(u_{i+1}^h)'(X_{i+1}) &= U_{i+1}\left(\frac{3h}{2h^2}\right) + U_i\left(\frac{-2h}{h^2}\right) + U_{i-1}\left(\frac{h}{2h^2}\right) \\
&= \frac{1}{h}\left(\frac{3}{2}U_{i+1} - 2U_i + \frac{1}{2}U_{i-1}\right)
\end{aligned}$$



Méthode de Gear d'ordre 2

$$U_{i+1} = \frac{4}{3}U_i - \frac{1}{3}U_{i-1} + \frac{2h}{3}F_{i+1}.$$

A quoi servent les méthodes numériques ?

Les artilleurs sont à l'origine de nombreuses méthodes de calcul moderne.

ou lorsque la vitesse et la précision de votre calcul pouvaient vous sauver la vie :-)

http://ww2ol.jeuxonline.info/article.php3?id_article=46

Plan du cours de méthodes numériques

Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Et les équations non-linéaires ?

Comment intégrer numériquement une fonction ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Problèmes non linéaires

Fonction non linéaire

Trouver x tel que

$f(x) = 0$

Suite de candidats…

$x_1, x_2, x_3, \ldots$

… qui devrait converger vers une racine

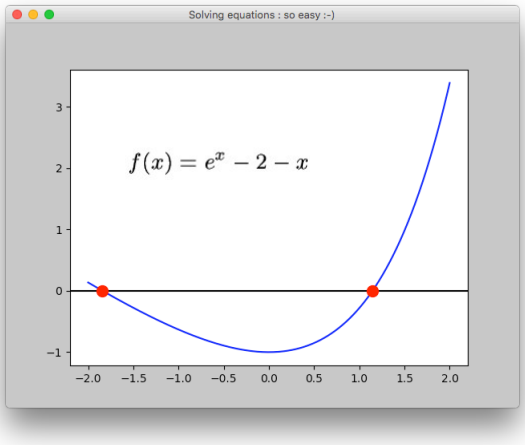
Questions

- Convergence vers une racine
- Candidat initial
- Encadrement

Méthodes numériques itératives

- Méthode de bissection
- Méthode du point fixe
- Méthode de Newton-Raphson

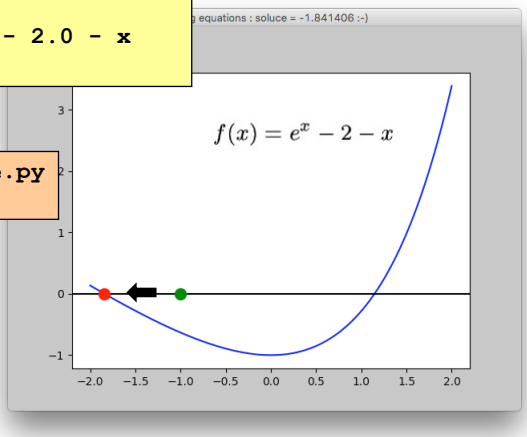
Exemple simple



Avec scipy, c'est très facile…

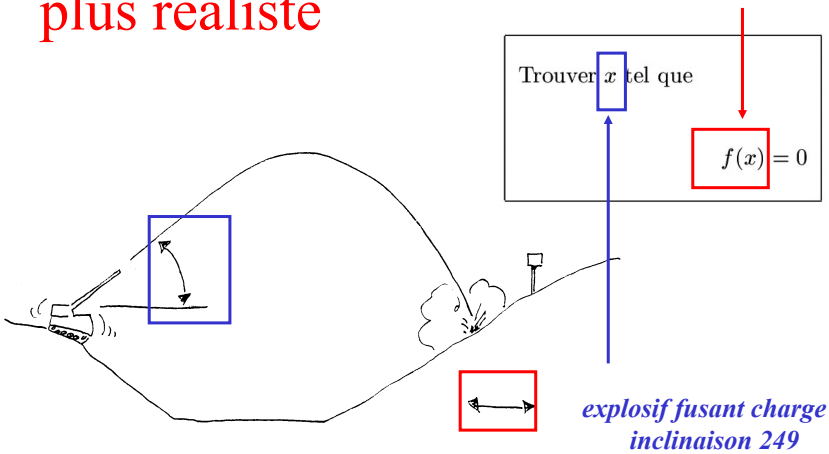
```
from math import exp
from scipy.optimize import fsolve

f = lambda x : exp(x) - 2.0 - x
print(fsolve(f,-1))
```

```
bash-3.2$ python solve.py
[-1.84140566]
```



Exemple plus réaliste

Distance entre l'explosion et la cible



explosif fusant charge 3
inclinaison 249
déplacement 4559

Comment prédire ou prévoir f ?

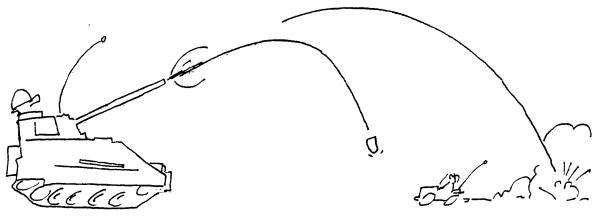
Trouver x tel que

$$f(x) = 0$$

Trajectoire de l'obus = solution d'une équation différentielle ordinaire
Paramètres à tenir en compte :
- ***Charge de poudre, type d'obus,***
- ***Usure du tube (change après chaque coup !)***
- ***Calibrage de l'obusier (différent pour chaque pièce !)***
- ***Position calculée de la pièce d'artillerie (travail de topographie)***
- ***Position estimée de la cible (observateur avancé)***
- ***Vents dominants, humidité de l'air***
- ***Rotation de la terre***

On obtient donc f(x) en utilisant ode45…

Evidemment entre modèle et réalité…



Méthodes itératives

C constante positive et inférieure à l'unité

$$\lim_{i\to\infty} \frac{|e_i|}{|e_{i-1}|^r} = C$$

$$e_i = x - x_i$$

taux de convergence

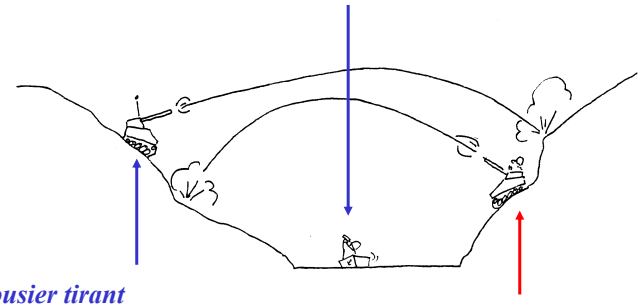
La séquence converge si on peut trouver C et r

r = taux de convergence

r=1 taux de convergence linéaire
r=2 taux de convergence quadratique

Est-il important de converger rapidement ?

Observateur avancé ajustant le tir (espérance de vie très limitée en général…)



Obusier tirant sur l'obusier adverse…

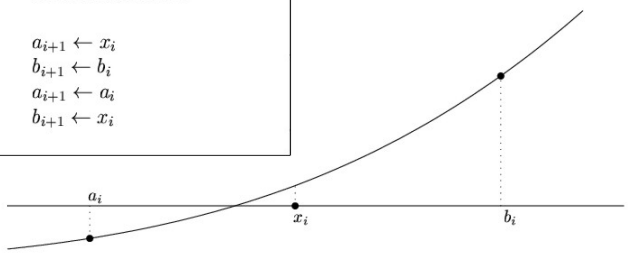
Obusier adverse tentant de vous détruire au même moment…

La technique de bissection est une méthode d'encadrement

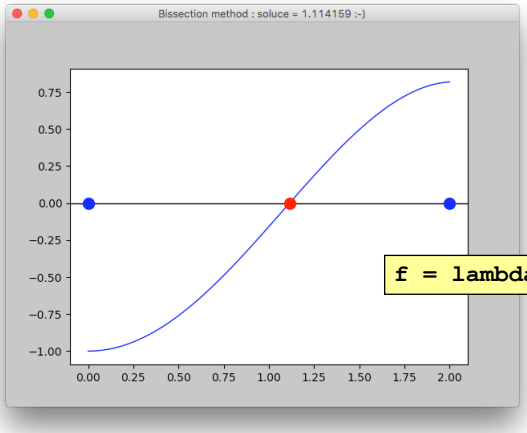
On fournit un intervalle $[a_0, b_0]$ avec $f(a_0)f(b_0) < 0$ et on calcule trois suites a_i, b_i, x_i par

$$x_i = \frac{(a_i + b_i)}{2}$$

Si $f(x_i) = 0$	On a une solution !
Si $f(x_i)f(a_i) > 0$	$a_{i+1} \leftarrow x_i$ $b_{i+1} \leftarrow b_i$
Sinon	$a_{i+1} \leftarrow a_i$ $b_{i+1} \leftarrow x_i$

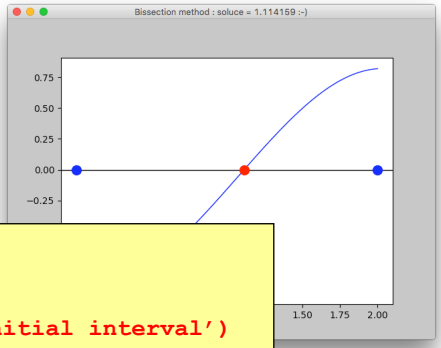


Exemple pour la méthode de bissection…




```
f = lambda x : x * sin(x) - 1
```

Programme python




```
def bissect(a,b,f,tol,nmax):
  n = 0; delta = (b-a)/2
  if (f(a)*f(b) > 0) :
    raise RuntimeError('Bad initial interval')
  while (abs(delta) >= tol and n <= nmax) :
    delta = (b-a)/2; n = n + 1;
    x = a + delta
    if (f(x)*f(a) > 0) :
      a = x
    else :
      b = x
  if (n > nmax) :
    raise RuntimeError('Too much iterations')
  return x
```

Une fioriture numérique…

```
x = a + (b-a)/2
```

```
>>> import numpy as np
>>> b = np.finfo(float).max
>>> a = b - 10**300
>>> (a+b)/2
inf
>>> a + (b-a)/2
1.7976931298623156e+308
>>> (b-a)/2
4.999999900185599e+299
```

…ou…

```
x = (a+b)/2
```

Et le résultat…

```
x =  1.0000000e+00 (Estimated error 1.0000000e+00 at iteration 1)
x =  1.5000000e+00 (Estimated error 5.0000000e-01 at iteration 2)
x =  1.2500000e+00 (Estimated error 2.5000000e-01 at iteration 3)
x =  1.1250000e+00 (Estimated error 1.2500000e-01 at iteration 4)
x =  1.0625000e+00 (Estimated error 6.2500000e-02 at iteration 5)
x =  1.0937500e+00 (Estimated error 3.1250000e-02 at iteration 6)
x =  1.1093750e+00 (Estimated error 1.5625000e-02 at iteration 7)
x =  1.1171875e+00 (Estimated error 7.8125000e-03 at iteration 8)
x =  1.1132812e+00 (Estimated error 3.9062500e-03 at iteration 9)
x =  1.1152344e+00 (Estimated error 1.9531250e-03 at iteration 10)
x =  1.1142578e+00 (Estimated error 9.7656250e-04 at iteration 11)
x =  1.1137695e+00 (Estimated error 4.8828125e-04 at iteration 12)
x =  1.1140137e+00 (Estimated error 2.4414062e-04 at iteration 13)
x =  1.1141357e+00 (Estimated error 1.2207031e-04 at iteration 14)
x =  1.1141968e+00 (Estimated error 6.1035156e-05 at iteration 15)
x =  1.1141663e+00 (Estimated error 3.0517578e-05 at iteration 16)
x =  1.1141510e+00 (Estimated error 1.5258789e-05 at iteration 17)
x =  1.1141586e+00 (Estimated error 7.6293945e-06 at iteration 18)
```

On observe un taux de convergence linéaire…

Taux de convergence linéaire

$$|e_0| \leq \frac{b_0 - a_0}{2}$$

En effectuant une nouvelle itération

$$|e_1| \leq \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2^2}$$

$$|e_n| \leq \frac{b_0 - a_0}{2^{n+1}}$$

$$|e_i| \leq \frac{1}{2}|e_{i-1}|$$

Problème aux conditions aux limites

Trouver $u(x)$ tel que

$$\begin{cases} u''(x) = f(x, u(x)), & x \in [a, b] \\ u(a) = u^a \\ u(b) = u^b \end{cases}$$

Trouver $(u(x), v(x))$ tels que

$$\begin{cases} u'(x) = v(x) \\ v'(x) = f(x, u(x)), & x \in [a, b] \\ u(a) = u^a \\ u(b) = u^b \end{cases}$$

Première idée : Méthode du tir

$r(v^a) = 0$

C'est la technique de l'artilleur

Technique d'ajustement par essais et erreurs… et une méthode d'encadrement

$$r : (v^a) \rightarrow u^b - u^h(b, v^a)$$

En d'autres mots…

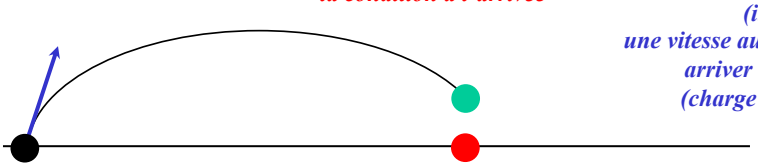
Trouver $(u(x), v(x))$ tels que

$$\begin{cases} u'(x) = v(x) \\ v'(x) = f(x, u(x)), \qquad x \in [a, b] \\ u(a) = u^a \\ u(b) = u^b \end{cases}$$

$$v(a) = v_0^a$$

(i) On oublie la condition à l'arrivée

(ii) On estime une vitesse au départ pour arriver au bon point (charge de poudre !)



(iii) On calcule la courbe avec Runge-Kutta à partir de u(a) et v(a) et on regarde où on arrive en u(b) !

$$r_0 = u^b - u_0^h(b)$$

Et on ajuste…

$$r_0 = u^b - u_0^h(b)$$

(iv) On adapte la quantité de poudre pour toucher la cible

v_i^a

$$r_1 = u^b - u_1^h(b)$$

On peut utiliser la méthode de bissection pour estimer la nouvelle vitesse à imposer pendant la procédure d'ajustage

$$r : (v^a) \rightarrow u^b - u^h(b, v^a)$$

Seconde idée : Différences finies

$$u(X_i) \approx u^h(X_i) = U_i$$

C'est la vision satellitaire

Synthèse globale des conditions limites par la résolution d'un système d'équations

$$0 = u''(X_i) - f(X_i, U_i) \approx (u^h)''(X_i) - f(X_i, U_i)$$

En utilisant une différence finie centrée d'ordre deux,

$$\approx \frac{(U_{i-1} - 2U_i + U_{i+1})}{h^2} - f(X_i, U_i)$$

Il est alors nécessaire de résoudre un système…

$$\frac{1}{h^2}\begin{bmatrix} h^2 & 0 & & & & & & & \\ 1 & -2 & 1 & & & & & & \\ & 1 & -2 & 1 & & & & & \\ & & 1 & -2 & 1 & & & & \\ & & & 1 & -2 & 1 & & & \\ & & & & \cdot & \cdot & \cdot & & \\ & & & & & 1 & -2 & 1 & \\ & & & & & & 1 & -2 & 1 \\ & & & & & & & 0 & h^2 \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ \vdots \\ U_{m-2} \\ U_{m-1} \\ U_m \end{bmatrix} = \begin{bmatrix} u^a \\ f(X_1, U_1) \\ f(X_2, U_2) \\ f(X_3, U_3) \\ f(X_4, U_4) \\ \vdots \\ \vdots \\ f(X_{m-1}, U_{m-1}) \\ u^b \end{bmatrix}$$

Système linéaire si f linéaire en u
Système non-linéaire si f non-linéaire en u

Exemple (linéaire)

$$u''(x) = x - u(x)$$

Deux constantes à choisir pour satisfaire les deux conditions aux limites !

$$u(x) = x + Acos(x) + Bsin(x)$$

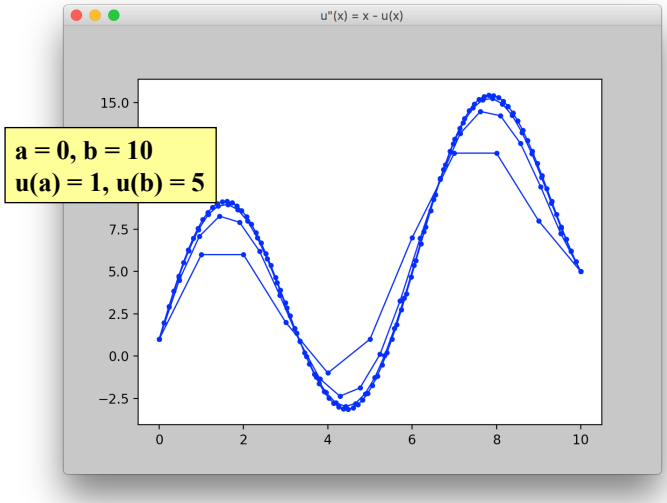
Solution analytique

Problème numérique

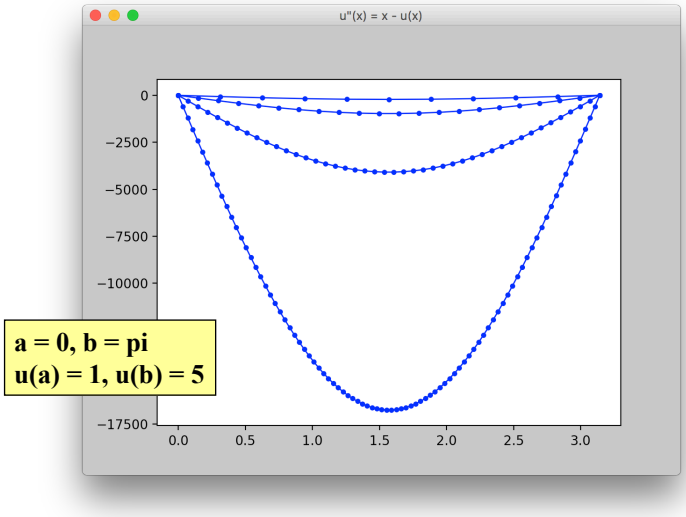
$$\frac{1}{h^2}\begin{bmatrix} 0 & 0 & & & & & & & \\ 1 & -2 & 1 & & & & & & \\ & 1 & -2 & 1 & & & & & \\ & & 1 & -2 & 1 & & & & \\ & & & 1 & -2 & 1 & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & 1 & -2 & 1 & \\ & & & & & & 1 & -2 & 1 \\ & & & & & & & 0 & 0 \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ \vdots \\ U_{m-2} \\ U_{m-1} \\ U_m \end{bmatrix} = \begin{bmatrix} u^a \\ X_1 \\ X_2 \\ X_3) \\ X_4 \\ \vdots \\ \vdots \\ X_{m-1} \\ u^b \end{bmatrix} - \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ \vdots \\ U_{m-2} \\ U_{m-1} \\ U_m \end{bmatrix}$$

```
X = linspace(a,b,n)
h = (b-a)/(n-1)

b = array([0,*ones(n-2),0])
A = spdiags([b,-2*b,b],[-1,0,1],n,n)/(h*h)
B = copy(X); B[0] = Ua; B[-1] = Ub

U = spsolve(A.T + eye(n,n),B)
```

Et hop, un joli résultat :-)

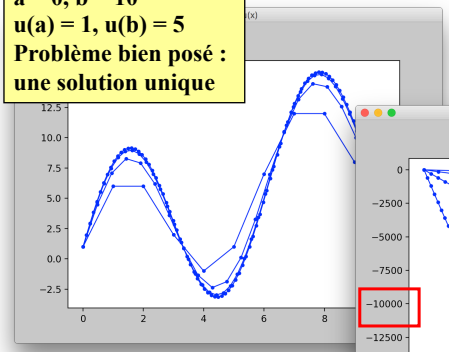


Un autre joli résultat :-(

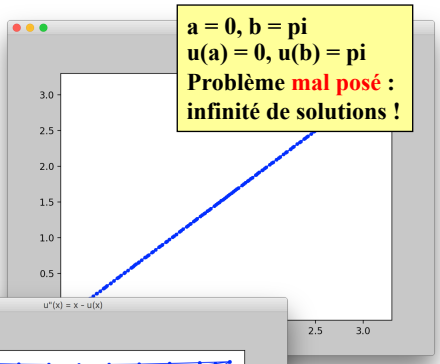


Est-ce que cela marche toujours ?

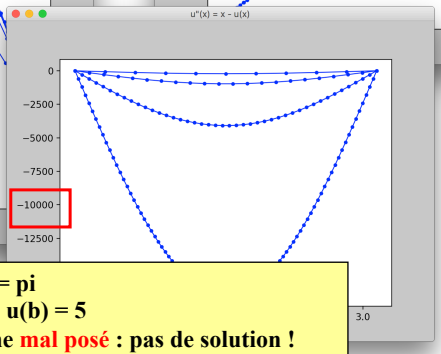
a = 0, b = 10
u(a) = 1, u(b) = 5
Problème bien posé : une solution unique



a = 0, b = pi
u(a) = 0, u(b) = pi
Problème mal posé : infinité de solutions !



a = 0, b = pi
u(a) = 1, u(b) = 5
Problème mal posé : pas de solution !
Les solutions obtenues n'ont aucun sens !



Problèmes non linéaires

Fonction non linéaire

Trouver x tel que

$$f(x) = 0$$

Suite de candidats…

$x_1, x_2, x_3, \ldots$

… qui devrait converger vers une racine

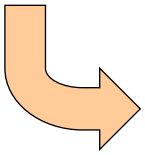
Questions
- Convergence vers une racine
- Candidat initial
- Encadrement

Méthodes numériques itératives
- Méthode de bissection
- Méthodes du point fixe
- Méthode de Newton-Raphson

Méthode du point fixe

Trouver x tel que

$$f(x) = 0$$



On reformule le problème…

Il existe plein de possibilités de choix pour g !

Trouver x tel que

$$x = g(x)$$

Méthode du point fixe

On fournit x_0

Tant que $|\Delta x| > \epsilon$, on calcule x_{i+1} à partir de x_i avec

$$x_{i+1} = g(x_i)$$

Si on converge, la solution x est le dernier x_{i+1} calculé

$$x_{i+1} = g(x_i)$$

Comment « bien » définir g à partir de f ?

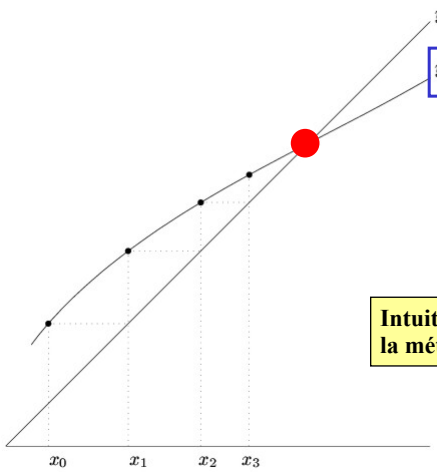
Réécrivons l'équation $f(x) = 0$ sous une forme[2]

[2]Il existe une infinité de façons d'écrire une équation $f(x) = 0$ sous une forme $x = g(x)$. A titre d'exemple considérons $f(x) = x^3 - 3x + 1$

On peut ainsi écrire $x = \frac{(x^3+1)}{3}$, ou $x = (3x-1)^{1/3}$ ou encore $x = x - x^3 + 3x - 1$...

C'est une bonne question…
On va s'y intéresser d'ici peu

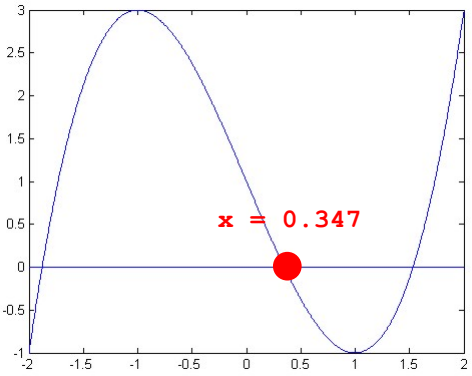
Interprétation géométrique



$$x_{i+1} = g(x_i)$$

Intuitivement, on observe que
la méthode converge si |g'(x)| < 1

Exemple

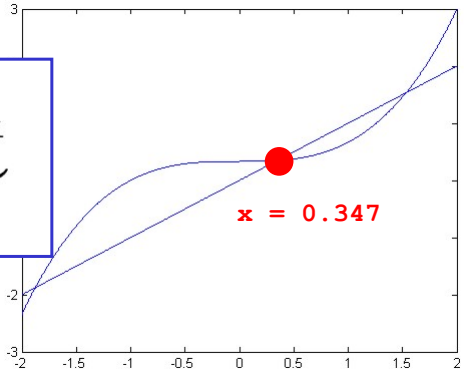


$$\underbrace{x^3 - 3x + 1}_{f(x)} = 0$$

Une itération possible…

$$\underbrace{x^3 - 3x + 1}_{f(x)} = 0$$

$$x_{i+1} = \underbrace{\frac{x_i^3 + 1}{3}}_{g(x_i)}$$



Une implémentation possible…

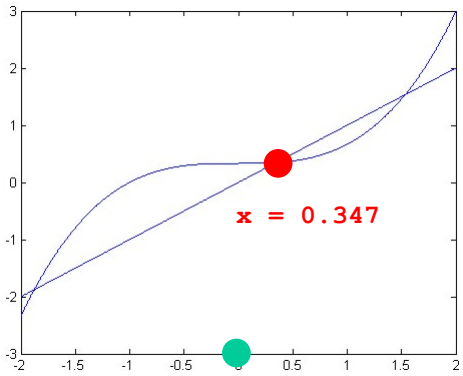
```
def iter(x,tol,nmax,g):
  n = 0; delta = float("inf")
  while abs(delta) > tol and n < nmax :
    n = n + 1
    xold = x
    x = g(x)
    delta = xold - x
    print(" x = %21.14e (%d)" % (x,n))
  return x
```

```
g = lambda x : (x**3 + 1)/3
```

$$x_{i+1} = \underbrace{\frac{x_i^3 + 1}{3}}_{g(x_i)}$$

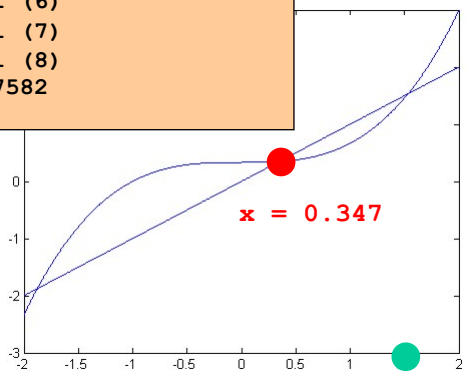
```
>>> print("Found x = ", iter(0.0,10e-3,50,g))
 x =  3.33333333333333e-01 (1)
 x =  3.45679012345679e-01 (2)
 x =  3.47102186947062e-01 (3)
Found x =  0.3471021869470616
```

Essayons…



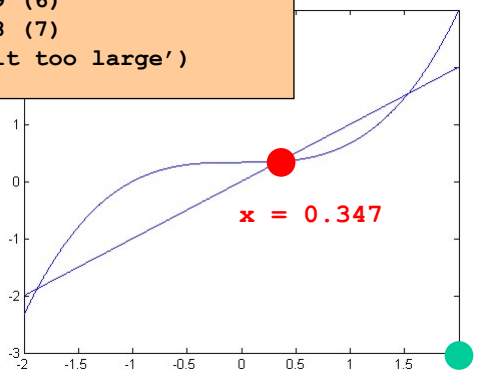
```
>>> print("Found x = ", iter(1.5,10e-3,50,g))
 x =  1.45833333333333e+00 (1)
 x =  1.36716338734568e+00 (2)
 x =  1.18513797762971e+00 (3)
 x =  8.88195982531111e-01 (4)
 x =  5.66896932292182e-01 (5)
 x =  3.94061625221864e-01 (6)
 x =  3.53730562615967e-01 (7)
 x =  3.48086882210758e-01 (8)
Found x =  0.3480868822107582
```

Et en partant d'un autre point…




```
>>> print("Found x = ", iter(2.0,10e-3,50,g))
 x =  3.00000000000000e+00 (1)
 x =  9.33333333333333e+00 (2)
 x =  2.71345679012346e+02 (3)
 x =  6.65959007564967e+06 (4)
 x =  9.84512506785963e+19 (5)
 x =  3.18084464276016e+59 (6)
 x = 1.07276876343287e+178 (7)
OverflowError: (34, 'Result too large')
```

Et encore plus loin…



Et de manière plus rigoureuse ?

Théorème 5.1. *Supposons que $g(x)$ et $g'(x)$ sont continues sur l'intervalle $[a, b]$ qui contient le point fixe unique x de la fonction g. Si la valeur de départ x_0 est choisie dans cet intervalle et si la condition suivante (dite condition de Lipschitz) est satisfaite*

$$|g'(x)| \leq K < 1 \qquad \forall x \in [a, b],$$

alors l'itération $x_{i+1} = g(x_i)$ converge vers x.

Pour les fonctions lipschitziennes…

$$\underbrace{(x_{i+1} - x)}_{e_{i+1}} = g(x_i) - g(x)$$

$\downarrow$ En vertu du théorème de la moyenne.

$$= g'(\xi)\ (x_i - x)$$

$\downarrow$ En vertu de la condition de Lipschitz.

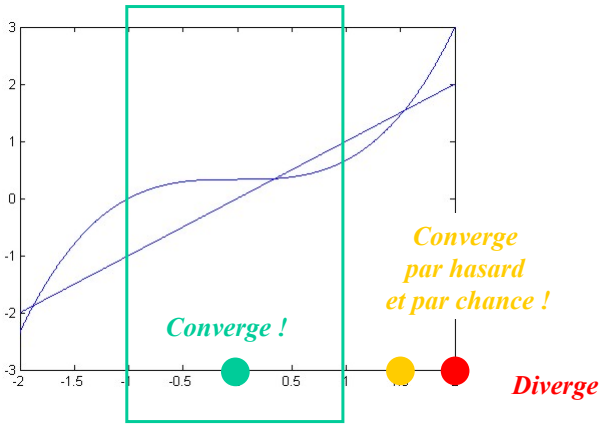
$$|e_{i+1}| \leq K\ |e_i|$$

$\Rightarrow$

$$0 \leq \lim_{i \to \infty} |x_i - x| \leq \lim_{i \to \infty} K^i\ |x_0 - x| = 0$$

Est-ce logique ?

$$x_{i+1} = \underbrace{\frac{x_i^3 + 1}{3}}_{g(x_i)}$$



Zone de convergence garantie : g'(x) = x^2 dans l'intervalle –1, 1 !

Méthode de Newton-Raphson

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2}f''(x_0) + \ldots$$

On fournit x_0

Tant que $|\Delta x| > \epsilon$, on calcule x_{i+1} à partir de x_i avec

$$f'(x_i)\overbrace{(x_{i+1} - x_i)}^{\Delta x} = -f(x_i)$$

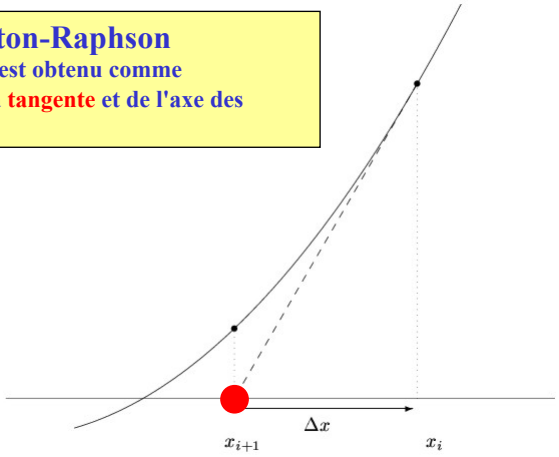
$$x_{i+1} = x_i + \Delta x$$

Si on converge, la solution x est le dernier x_{i+1} calculé

Interprétation géométrique

Méthode de Newton-Raphson

Le nouveau point est obtenu comme l'intersection de la tangente et de l'axe des abscisses



Newton-Raphson : Taux de convergence quadratique

Propagation de l'erreur dans un schéma de Newton-Raphson

$$\begin{aligned} e_{i+1} &= x - x_{i+1} \\ &= x - \left(x_i - \frac{f(x_i)}{f'(x_i)} \right) \\ &= \frac{e_i f'(x_i) + f(x_i)}{f'(x_i)} \end{aligned}$$

Développement en série de Taylor

$$\begin{aligned} 0 &= f(x) \\ &= f(x_i) + \underbrace{(x - x_i)}_{e_i} f'(x_i) + \underbrace{(x - x_i)^2}_{e_i^2} \frac{f''(\xi)}{2} \end{aligned}$$

$$e_{i+1} = \underbrace{-\frac{1}{2}\frac{f''(\xi)}{f'(x_i)}}_{C} e_i^2$$

Convergence si cette constante est comprise entre [-1,1]…

$$f'(x_i) \approx \frac{f(x_i+h) - f(x_i-h)}{2h}$$

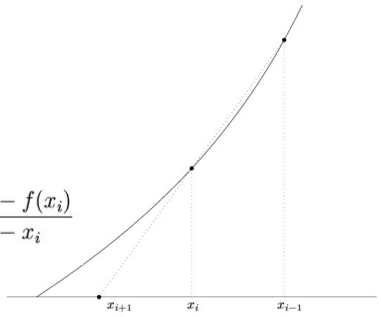
Evaluation numérique de f'

Deux estimations de f requises.
Difficulté de sélectionner h…

Une idée particulière

Une seule estimation de f requise.
Pas de paramètre à choisir !

$$f'(x_i) \approx \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$



Méthode de la sécante

$$|e_{i+1}| = C|e_i|^{1.618}$$

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

Quelle est la méthode qui converge le plus rapidement ?

Taux de convergence quadratique

$$e_{i+1} = -\underbrace{\frac{1}{2}\frac{f''(\xi)}{f'(x_i)}}_{C} e_i^2$$

Méthode de Newton-Raphson
1 itération revient à calculer
1 estimation de f
1 estimation de f'

Taux de convergence superlinéaire mais pas quadratique !

$$|e_{i+1}| = C|e_i|^{1.618}$$

Méthode de la sécante
1 itération revient à calculer
1 estimation de f

$1.618^2 = 2.6179 > 2$

Systèmes d'équations non-linéaires

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ \ldots & \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{cases}$$

Notation compacte : les vecteurs sont en gras.

$$\mathbf{f}(\mathbf{x}) = 0$$

Que peut-on faire avec les systèmes ?

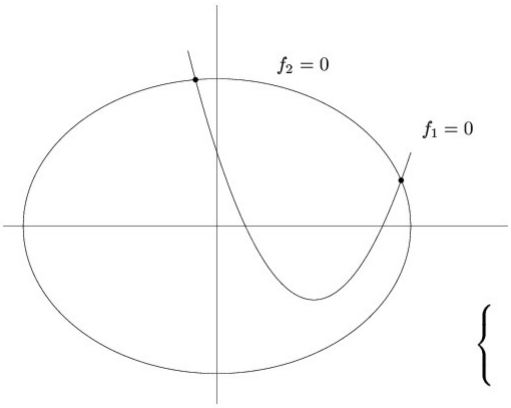
Méthodes numériques itératives
- **Méthode de bissection**
- **Méthodes du point fixe**
- **Méthode de Newton-Raphson**

Robuste, converge toujours si on a un intervalle de départ !

Mais pas généralisable aux systèmes !

Généralisables de manière immédiate aux systèmes..
Ne convergent que sous conditions…
Nécessitent un candidat initial proche de la solution…

Trouver la solution d'un système non-linéaire est très très difficile !



$$\begin{cases} \overbrace{x_1^2 - 2x_1 - x_2 + 0.5}^{f_1(x_1,x_2)} = 0 \\ \underbrace{x_1^2 + 4x_2^2 - 4}_{f_2(x_1,x_2)} = 0 \end{cases}$$

Méthode du point fixe

On fournit $\mathbf{x}_0$

Tant que $\| \Delta \mathbf{x} \| > \epsilon$, on calcule $\mathbf{x}_{i+1}$ à partir de $\mathbf{x}_i$ avec

$$\mathbf{x}_{i+1} = \mathbf{g}(\mathbf{x}_i)$$

Si on converge, la solution $\mathbf{x}$ est le dernier $\mathbf{x}_{i+1}$ calculé

Notation compacte : les vecteurs sont en gras.

Condition de Lipschitz

$$\sum_{i=1}^{n} \left| \frac{\partial g_j}{\partial x_i} \right| < 1 \qquad j = 1 \ldots n$$

La méthode du point fixe convergera vers la racine si la condition de Lipschitz est satisfaite !

Méthode du point fixe

$$\begin{cases} x_{1,i+1} = \dfrac{x_{1,i}^2 - x_{2,i} + 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i}^2 - 4x_{2,i}^2 + 8x_{2,i} + 4}{8} \end{cases}$$

Itération de Paul

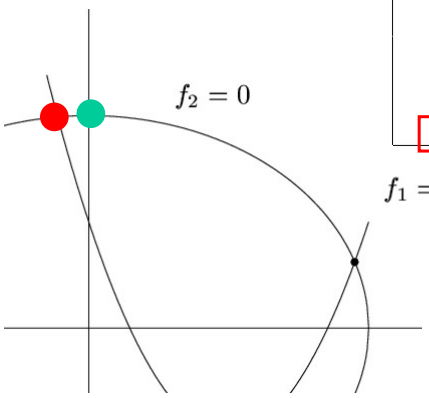
$$\begin{cases} \overbrace{x_1^2 - 2x_1 - x_2 + 0.5}^{f_1(x_1,x_2)} = 0 \\ \underbrace{x_1^2 + 4x_2^2 - 4}_{f_2(x_1,x_2)} = 0 \end{cases}$$

$$\begin{cases} x_{1,i+1} = \dfrac{-x_{1,i}^2 + 4x_{1,i} + x_{2,i} - 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i}^2 - 4x_{2,i}^2 + 11x_{2,i} + 4}{11} \end{cases}$$

Itération de Pierre

Parfois, cela marche…

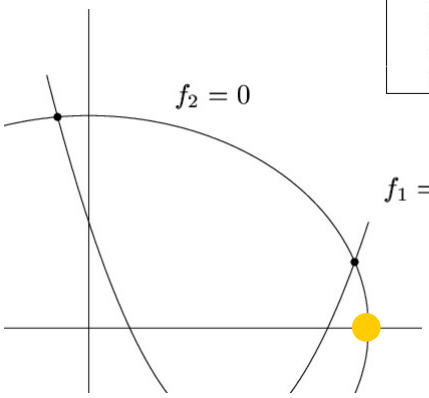
i	$x_{1,i}$	$x_{2,i}$
0	0.0000000	1.0000000
1	- 0.2500000	1.0000000
2	- 0.2187500	0.9921875
3	- 0.2221680	0.9939880
4	- 0.2223147	0.9938121
5	- 0.2221941	0.9938029
6	- 0.2222163	0.9938095
7	- 0.2222147	0.9938083
8	- 0.2222145	0.9938084
9	- 0.2222146	0.9938084



$$\begin{cases} x_{1,i+1} = \dfrac{x_{1,i}^2 - x_{2,i} + 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i}^2 - 4x_{2,i}^2 + 8x_{2,i} + 4}{8} \end{cases}$$

Parfois, cela diverge..

i	$x_{1,i}$	$x_{2,i}$
0	2.0000000	0.0000000
1	2.2500000	0.0000000
2	2.7812500	- 0.1328125
3	4.1840820	- 0.6085510
4	9.3075467	- 2.4820360
5	44.8062311	- 15.8910907
6	1 011.9947186	- 392.6042650
7	512263.2073904	-205477.8225378

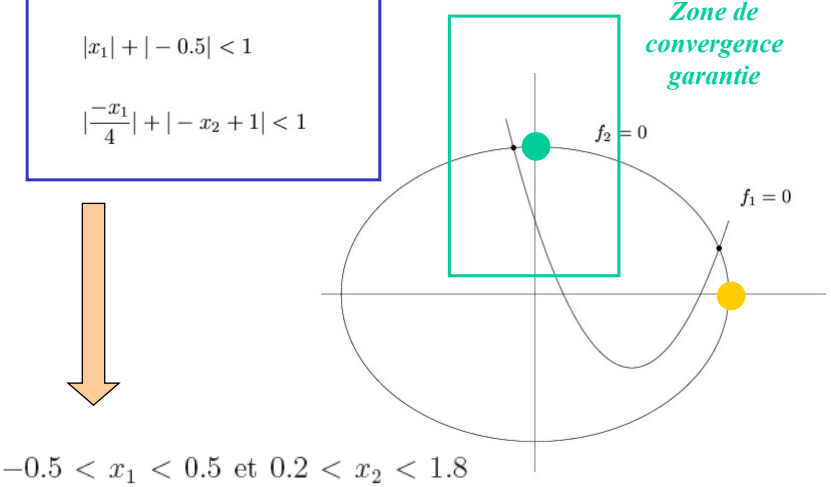


$$\begin{cases} x_{1,i+1} = \dfrac{x_{1,i}^2 - x_{2,i} + 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i}^2 - 4x_{2,i}^2 + 8x_{2,i} + 4}{8} \end{cases}$$

Et la condition de Lipschitz…

$$|x_1| + |-0.5| < 1$$

$$|\frac{-x_1}{4}| + |-x_2 + 1| < 1$$

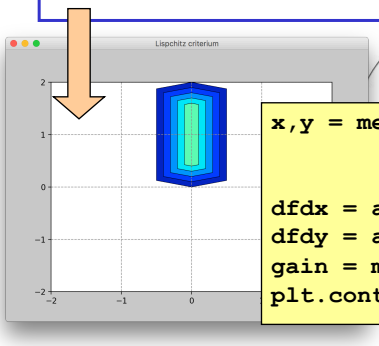


$-0.5 < x_1 < 0.5$ et $0.2 < x_2 < 1.8$

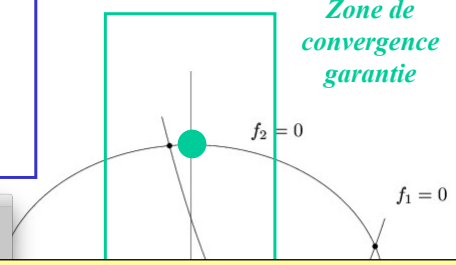
Et la condition de Lipschitz…

$$|x_1| + |-0.5| < 1$$

$$|\frac{-x_1}{4}| + |-x_2 + 1| < 1$$



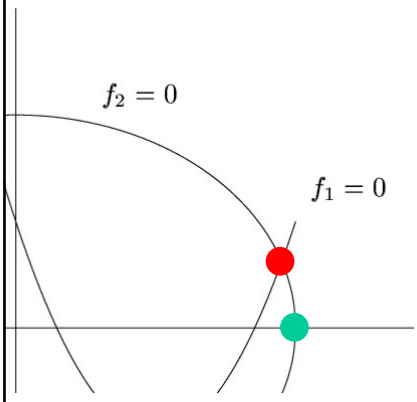
Zone de convergence garantie




```
x,y = meshgrid(linspace(-2,2,1000),
               linspace(-2,2,1000))

dfdx = abs(x)    + 0.5
dfdy = abs(x/4) + abs(-y+1)
gain = maximum(dfdx,dfdy)
plt.contourf(x,y,gain,arange(0,1.1,0.1)
```

Une autre itération converge...

$$\begin{cases} x_{1,i+1} = \dfrac{-x_{1,i}^2 + 4x_{1,i} + x_{2,i} - 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i}^2 - 4x_{2,i}^2 + 11x_{2,i} + 4}{11} \end{cases}$$



i	$x_{1,i}$	$x_{2,i}$
0	2.0000000	0.0000000
1	1.7500000	0.0000000
2	1.7187500	0.0852273
3	1.7530629	0.1776676
4	1.8083448	0.2504410
8	1.9035947	0.3160782
12	1.9009241	0.3112267
16	1.9006516	0.3111994
20	1.9006771	0.3112196
24	1.9006768	0.3112185

Est-il possible d'améliorer la vitesse de convergence ?

$$\begin{cases} x_{1,i+1} = \dfrac{-x_{1,i}^2 + 4x_{1,i} + x_{2,i} - 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i}^2 - 4x_{2,i}^2 + 11x_{2,i} + 4}{11} \end{cases}$$

$$\begin{cases} x_{1,i+1} = \dfrac{-x_{1,i}^2 + 4x_{1,i} + x_{2,i} - 0.5}{2} \\ x_{2,i+1} = \dfrac{-x_{1,i+1}^2 - 4x_{2,i}^2 + 11x_{2,i} + 4}{11} \end{cases}$$

Algorithme de Seidel
On utilise la dernière valeur disponible pour chaque inconnue
Parfois, cela converge plus vite,
Parfois, cela converge moins vite, parfois cela diverge…

Peut-on appliquer la méthode du point fixe à un système linéaire ?

$$\underbrace{\mathbf{Ax} - \mathbf{b}}_{\mathbf{f(x)}} = 0$$

Méthodes itératives (pt fixe)
- Parfois plus rapides
- Mais, ne convergent pas toujours
- Moins gourmandes en mémoire
- Utiles pour les très grands systèmes

Méthodes directes
- Elimination gaussienne (technique d'échelonnement du cours d'algèbre)
- Résultat toujours obtenu en un nombre fini d'opérations
- Nombre d'opérations requises = n^3

Méthodes
de Jacobi et de Gauss-Seidel

$$\underbrace{\mathbf{A}\mathbf{x} - \mathbf{b}}_{\mathbf{f}(\mathbf{x})} = 0$$

$$\mathbf{x}_{i+1} = \underbrace{\mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})\mathbf{x}_i + \mathbf{D}^{-1}\mathbf{b}}_{\mathbf{g}_{Jacobi}(\mathbf{x}_i)}$$

Approximation de A^{-1}

$$\mathbf{x}_{i+1} = \underbrace{(\mathbf{D}+\mathbf{L})^{-1}}\ \overbrace{(\mathbf{D}+\mathbf{L}-\mathbf{A})}^{-\mathbf{U}}\ \mathbf{x}_i + \underbrace{(\mathbf{D}+\mathbf{L})^{-1}\ \mathbf{b}}_{\mathbf{g}_{Gauss-Seidel}(\mathbf{x}_i)}$$

Exemple concret

$$\begin{bmatrix} 5 & 3 \\ 4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

On remplace la deuxième équation par « equation(2) – 2 equation(1) » pour obtenir la convergence !

```
def g(x):
  y = copy(x)
  y[0] =  (-3*x[1]+6)/5
  y[1] =  -(6*x[0]-4)/8
  return y

def jacobi(x,tol,nmax):
  n = 0; delta = tol+1;
  x = array(x,dtype=float)
  while (norm(delta) > tol and n < nmax):
    n = n+1; xold = x.copy()
    x = g(x)
    delta = x - xold
  return x

print(jacobi([0,0],10e-6,50))
```

Et Gauss-Seidel ?

$$\begin{bmatrix} 5 & 3 \\ 4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

L'implémentation de Gauss-Seidel est plus simple que celle de Jacobi !

```
def g(x):
  y = copy(x)
  y[0] =  (-3*x[1]+6)/5
  y[1] =  -(6*
  return y

def jacobi(x,t
  n = 0; delta
  x = array(x,
  while (norm(
    n = n+1; x
    x = g(x)
    delta = x
  return x
```

```
def g(x):

  x[0] =  (-3*x[1]+6)/5
  x[1] =  -(6*x[0]-4)/8
  return x

def gaussseidel(x,tol,nmax):
  n = 0; delta = tol+1;
  x = array(x,dtype=float)
  while (norm(delta) > tol and n < nmax):
    n = n+1; xold = x.copy()
    x = g(x)
    delta = x - xold
return x
```

Jacobi

```
>>>print(jacobi([0,0],10e-6,50))
 Estimated error  1.3000000e+00 at iteration 1
 Estimated error  9.4868330e-01 at iteration 2
 Estimated error  5.8500000e-01 at iteration 3
 Estimated error  4.2690748e-01 at iteration 4
 Estimated error  2.6325000e-01 at iteration 5


 Estimated error  2.9436348e-05 at iteration 28
 Estimated error  1.8151752e-05 at iteration 29
 Estimated error  1.3246357e-05 at iteration 30
 Estimated error  8.1682883e-06 at iteration 31
[ 1.63636089 -0.72726502]
```

Gauss-Seidel

```
>>>print(gaussseidel([0,0],10e-6,50))
 Estimated error  1.2649111e+00 at iteration 1
 Estimated error  3.0000000e-01 at iteration 2
 Estimated error  1.3500000e-01 at iteration 3


 Estimated error  1.0215189e-04 at iteration 12
 Estimated error  4.5968349e-05 at iteration 13
 Estimated error  2.0685757e-05 at iteration 14
 Estimated error  9.3085907e-06 at iteration 15
[ 1.63635754 -0.72726816]
```

Condition pour qu'une méthode itérative converge ?

$$\underbrace{(\mathbf{x}_{i+1} - \mathbf{x})}_{\mathbf{e}_{i+1}} = \mathbf{M}\mathbf{x}_i + \mathbf{c} - \mathbf{M}\mathbf{x} - \mathbf{c}$$

$$\mathbf{x}_{i+1} = \mathbf{M}\mathbf{x}_i + \mathbf{c}$$

$$= \mathbf{M}\,(\mathbf{x}_i - \mathbf{x})$$

En procédant de la même manière pour chaque étape,

$$= \mathbf{M}^{i+1}\,\underbrace{(\mathbf{x}_0 - \mathbf{x})}_{\mathbf{e}_0}$$

Il faut que…

$$\lim_{i \to \infty} \mathbf{M}^i\,\mathbf{e}_0 = 0$$

Ecrivons l'erreur comme une combili de vecteurs propres…

$$\mathbf{e}_0 = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j$$

↓ Puisque $\mathbf{M}\ \mathbf{v}_i = \lambda_i \mathbf{v}_i$,

$$\mathbf{e}_i = \sum_{j=1}^{n} \alpha_j \lambda_j^i \mathbf{v}_j$$

Pour obtenir…

$$\lim_{i \to \infty} \mathbf{M}^i \mathbf{e}^{(0)} = 0$$

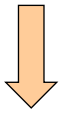
… on doit exiger que le rayon spectral de la matrice M soit inférieur à l'unité

$$|\lambda_i| < 1 \qquad \forall\ i$$

Condition pratique pour Jacobi

$$\sum_{j=1, \neq i}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| < 1 \qquad i = 1, \ldots, n$$

La dominance diagonale permet d'avoir la convergence…



Soit $\mathbf{M}$ *une matrice diagonalisable d'ordre n, on a alors :*

$$|\lambda_i| \leq \sum_{j=1}^{n} |m_{ij}| \qquad i = 1, \ldots, n$$

$$|\lambda_i| < 1 \qquad \forall\ i$$

Méthode de Newton-Raphson

On fournit $\mathbf{x}_0$

Tant que $\Delta\mathbf{x} > \epsilon$, on calcule $\mathbf{x}_{i+1}$ à partir de $\mathbf{x}_i$ avec

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_i)\overbrace{(\mathbf{x}_{i+1} - \mathbf{x}_i)}^{\Delta\mathbf{x}} = -\mathbf{f}(\mathbf{x}_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$$

Si on converge, la solution $\mathbf{x}$ est le dernier $\mathbf{x}_{i+1}$ calculé

Notation compacte : les vecteurs sont en gras.

Matrice jacobienne du système

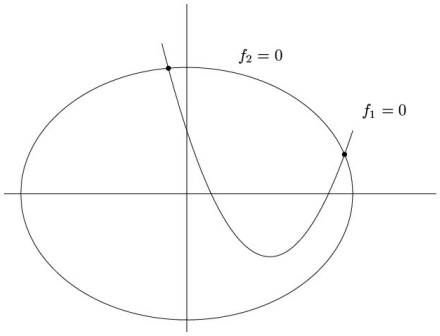
$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_i) = \left.\frac{\partial f_j}{\partial x_k}\right|_{(x_{1,i}, x_{2,i}, \ldots, x_{n,i})}$$

Tableau à deux indices
On calcule la dérivée partielle par rapport à la k-ième inconnue de la j-ième composante de f

Tableau à deux indices
Il s'agit de la seconde composante du vecteur Inconnue lors de la i-ème itération

Exemple

$$\begin{cases} \overbrace{x_1^2 - 2x_1 - x_2 + 0.5}^{f_1(x_1,x_2)} = 0 \\ \underbrace{x_1^2 + 4x_2^2 - 4}_{f_2(x_1,x_2)} = 0 \end{cases}$$



$$\left.\frac{\partial f_j}{\partial x_k}\right|_{(x_1,x_2)} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{(x_1,x_2)} & \left.\frac{\partial f_1}{\partial x_2}\right|_{(x_1,x_2)} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{(x_1,x_2)} & \left.\frac{\partial f_2}{\partial x_2}\right|_{(x_1,x_2)} \end{bmatrix}$$

$$= \begin{bmatrix} 2x_1 - 2 & -1 \\ 2x_1 & 8x_2 \end{bmatrix}$$

Comment appliquer la méthode de Newton-Raphson à ce système d'équations non-linéaires ?

On fournit $\mathbf{x}_0$

Tant que $\Delta\mathbf{x} > \epsilon$, on calcule $\mathbf{x}_{i+1}$ à partir de $\mathbf{x}_i$ avec

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_i)\overbrace{(\mathbf{x}_{i+1} - \mathbf{x}_i)}^{\Delta\mathbf{x}} = -\mathbf{f}(\mathbf{x}_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$$

Si on converge, la solution $\mathbf{x}$ est le dernier $\mathbf{x}_{i+1}$ calculé

$$\begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{(x_{1,i},x_{2,i})} & \left.\frac{\partial f_1}{\partial x_2}\right|_{(x_{1,i},x_{2,i})} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{(x_{1,i},x_{2,i})} & \left.\frac{\partial f_2}{\partial x_2}\right|_{(x_{1,i},x_{2,i})} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -f_1(x_{1,i}, x_{2,i}) \\ -f_2(x_{1,i}, x_{2,i}) \end{bmatrix}$$

Que fournit la méthode de Newton-Raphson pour ce système d'équations non-linéaires ?

$$\begin{cases} \overbrace{x_1^2 - 2x_1 - x_2 + 0.5}^{f_1(x_1,x_2)} &= 0 \\ \underbrace{x_1^2 + 4x_2^2 - 4}_{f_2(x_1,x_2)} &= 0 \end{cases}$$

On fournit $\mathbf{x}_0$

Tant que $\Delta\mathbf{x} > \epsilon$, on calcule $\mathbf{x}_{i+1}$ à partir de $\mathbf{x}_i$ avec

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_i)\overbrace{(\mathbf{x}_{i+1} - \mathbf{x}_i)}^{\Delta\mathbf{x}} = -\mathbf{f}(\mathbf{x}_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$$

Si on converge, la solution $\mathbf{x}$ est le dernier $\mathbf{x}_{i+1}$ calculé

i	$x_{1,i}$	$x_{2,i}$
0	2.000000	0.250000
1	1.906250	0.312500
2	1.900691	0.311213
4	1.900677	0.311219

Dans le domaine de convergence asymptotique, on double le nombre de chiffres significatifs à chaque itération

Exemple 1 : Optimisation non-linéaire

$$u^h(x) = \frac{a}{x+b}$$

$$f(a,b) = \sum_{i=1}^{n} \left(U_i - \frac{a}{(X_i+b)} \right)^2$$

Fonction à minimiser

n mesures (X_i, U_i)

On ajuste les deux paramètres *a* et *b* afin de minimiser le carré des écarts

Extréma d'une fonction à deux variables…

$$0 = \frac{\partial f}{\partial a}$$

$$0 = \frac{\partial f}{\partial b}$$

Conditions nécessaires pour obtenir un minimum !

Deux équations non-linéaires

$$0 = \sum_{i=1}^{n} 2 \left(U_i - \frac{a}{(X_i+b)} \right) \frac{-1}{(X_i+b)}$$

$$0 = \sum_{i=1}^{n} 2 \left(U_i - \frac{a}{(X_i+b)} \right) \frac{a}{(X_i+b)^2}$$

Utilisons Newton-Raphson !

On calcule (a_{k+1}, b_{k+1}) à partir de (a_k, b_k) avec

$$\begin{bmatrix} \frac{\partial^2 f}{\partial a^2}(a_k,b_k) & \frac{\partial^2 f}{\partial a \partial b}(a_k,b_k) \\ \frac{\partial^2 f}{\partial b \partial a}(a_k,b_k) & \frac{\partial^2 f}{\partial b^2}(a_k,b_k) \end{bmatrix} \begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix} = - \begin{bmatrix} \frac{\partial f}{\partial a}(a_k,b_k) \\ \frac{\partial f}{\partial b}(a_k,b_k) \end{bmatrix}$$

$$\begin{aligned} a_{k+1} &= a_k + \Delta a \\ b_{k+1} &= b_k + \Delta b \end{aligned}$$

$$\frac{\partial^2 f}{\partial a^2} = 2\sum_{i=1}^{n} \frac{1}{(X_i+b)^2}$$

$$\frac{\partial^2 f}{\partial b^2} = -4a\sum_{i=1}^{n} \frac{U_i}{(X_i+b)^3} + 6a^2 \sum_{i=1}^{n} \frac{1}{(X_i+b)^4}$$

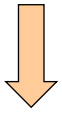
$$\frac{\partial^2 f}{\partial a \partial b} = \frac{\partial^2 f}{\partial b \partial a} = 2\sum_{i=1}^{n} \frac{U_i}{(X_i+b)^2} - 4a\sum_{i=1}^{n} \frac{1}{(X_i+b)^3}$$

$$\frac{\partial f}{\partial a} = -2\sum_{i=1}^{n} \frac{U_i}{(X_i+b)} + 2a\sum_{i=1}^{n} \frac{1}{(X_i+b)^2}$$

$$\frac{\partial f}{\partial b} = 2a\sum_{i=1}^{n} \frac{U_i}{(X_i+b)^2} - 2a^2\sum_{i=1}^{n} \frac{1}{(X_i+b)^3}$$

Et espérer que cela converge…

Il y aura un problème…. si on obtient une valeur de *b* identique à l'opposé des données !

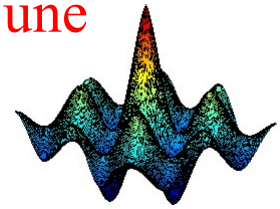


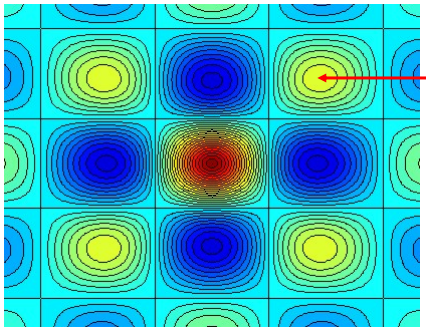
```
Warning: Divide by zero.
ans = Inf
```

$$\sum_{i=1}^{n} 2\left(U_i - \frac{a}{(X_i+b)}\right)\frac{-1}{(X_i+b)}$$

$$\sum_{i=1}^{n} 2\left(U_i - \frac{a}{(X_i+b)}\right)\frac{a}{(X_i+b)^2}$$

Trouver le maximum d'une fonction non-linéaire est une tâche très très difficile…





Un maximum local ?

Comment savoir la présence ou l'absence d'une fosse plus profonde sur base d'informations locales….

Impossible ?

Minimiser une fonction non-linéaire est très très difficile !

Le message est que rien n'est su de manière sûre. Il y a des choses dont nous savons que nous les savons. Il y a des inconnues connues, c'est-à-dire des choses dont nous savons maintenant que nous ne les savons pas.

Mais, il y a aussi des inconnues inconnues. Il y a des choses dont nous savons que nous ne les savons pas …. Et chaque année, nous découvrons un peu plus de ces inconnues inconnues.

… Il y a une autre façon de dire cela, c'est que l'inexistence de preuves n'est pas une preuve d'inexistence

Secrétaire à la Défense US à l'OTAN à Bruxelles, juin 2002 !

(Lewis Lapham)

Les méthodes numériques… existantes

Des propos dignes de l'un de ces fous énigmatiques qui hantent les forêts enchantées dans les pièces de Shakespeare

Exemple 2 : Systèmes d'équations différentielles

$$\begin{cases} u_1'(x) &= f_1(x, u_1(x), u_2(x), \ldots, u_n(x)) \\ u_2'(x) &= f_2(x, u_1(x), u_2(x), \ldots, u_n(x)) \\ \vdots & \\ u_n'(x) &= f_n(x, u_1(x), u_2(x), \ldots, u_n(x)) \end{cases}$$

$$\begin{cases} u_1(a) &= \overline{u}_1 \\ u_2(a) &= \overline{u}_2 \\ \vdots & \\ u_n(a) &= \overline{u}_n \end{cases}$$

Notation compacte : les vecteurs sont en gras.

Trouver $\mathbf{u}(x)$ tel que

$$\begin{cases} \mathbf{u}'(x) &= \mathbf{f}(x, \mathbf{u}(x)), \qquad x \in [a, b] \\ \mathbf{u}(a) &= \overline{\mathbf{u}} \end{cases}$$

C'est exactement la même histoire !

$$U_{ji} = u_j^h(X_i) \approx u_j(X_i)$$

$$\mathbf{U}_i = \mathbf{u}^h(X_i) \approx \mathbf{u}(X_i)$$

Notation compacte : les vecteurs sont en gras.

Méthode d'Euler explicite

$$U_{ji+1} = U_{ji} + h f_j(X_i, U_{1i}, \ldots, U_{ni})$$

Tableau à deux indices
On calcule la j-ème composante du vecteur des inconnues à la i-ème abscisse temporelle

Vecteur
On calcule la j-ème composante du vecteur des inconnues.

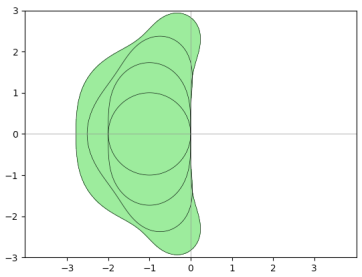
Stabilité des systèmes

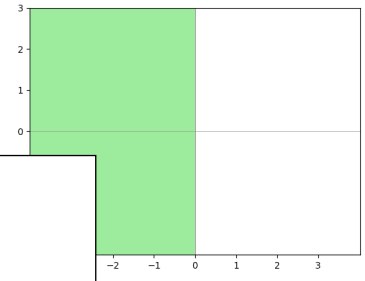
$\mathbf{u}' = \mathbf{A}\mathbf{u}$

$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$

$\mathbf{v}(x) = \mathbf{P}^{-1}\mathbf{u}(x)$

$\mathbf{v}' = \mathbf{D}\mathbf{v}$





Le problème différentiel initial est équivalent à n équations scalaires

$$v_i'(x) = \lambda_i v_i(x), \qquad i = 1, \ldots, n$$

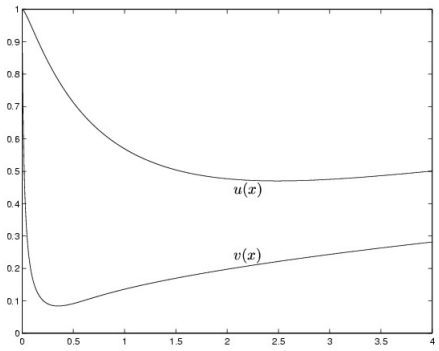
Système d'équations différentielles non-linéaires

Trouver $(u(x), v(x))$ tels que

$$\begin{cases} u'(x) &= -u^2(x) + v(x) \\ v'(x) &= -50\ v^2(x) + x, \qquad x \in [0,4] \\ \\ u(0) &= 1 \\ v(0) &= 1 \end{cases}$$

$$\begin{bmatrix} \left.\frac{\partial f_1}{\partial u}\right|_x & \left.\frac{\partial f_1}{\partial v}\right|_x \\ \left.\frac{\partial f_2}{\partial u}\right|_x & \left.\frac{\partial f_2}{\partial v}\right|_x \end{bmatrix} = \begin{bmatrix} -2u & 1 \\ 0 & -100v \end{bmatrix}$$

Il s'agit d'un problème raide



$$\det \begin{bmatrix} -2u(x) - \lambda & 1 \\ 0 & -100\ v(x) - \lambda \end{bmatrix} = 0$$

$$\downarrow$$

$$\begin{aligned} \lambda_1(x) &= -100\ v(x) \\ \lambda_2(x) &= -2\ u(x) \end{aligned}$$

Euler explicite

$$\begin{cases} U_{i+1} = U_i + h\ (-U_i^2 + V_i) \\ V_{i+1} = V_i + h\ (-50\ V_i^2 + X_i) \end{cases}$$

$$\begin{cases} U_{i+1} = U_i + h\ (-U_{i+1}^2 + V_{i+1}) \\ V_{i+1} = V_i + h\ (-50\ V_{i+1}^2 + X_{i+1}) \end{cases}$$

Euler implicite

Newton-Raphson et Euler implicite

$$\begin{cases} \overbrace{U_{i+1} - U_i - h\ \left(-U_{i+1}^2 + V_{i+1}\right)}^{g_1(U_{i+1}, V_{i+1})} &= 0 \\ \underbrace{V_{i+1} - V_i - h\ \left(-50\ V_{i+1}^2 + X_{i+1}\right)}_{g_2(U_{i+1}, V_{i+1})} &= 0 \end{cases}$$

Notation compacte : les vecteurs sont en gras.

$$\mathbf{g}(\mathbf{x}) = 0$$

Comment appliquer la méthode de Newton-Raphson à ce système d'équations non-linéaires ?

On fournit $\mathbf{x}_0$

Tant que $\Delta\mathbf{x} > \epsilon$, on calcule $\mathbf{x}_{i+1}$ à partir de $\mathbf{x}_i$ avec

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_i)\overbrace{(\mathbf{x}_{i+1} - \mathbf{x}_i)}^{\Delta \mathbf{x}} = -\mathbf{f}(\mathbf{x}_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$$

Si on converge, la solution $\mathbf{x}$ est le dernier $\mathbf{x}_{i+1}$ calculé

$$\begin{bmatrix} \left.\frac{\partial g_1}{\partial x_1}\right|_{(x_1,x_2)} & \left.\frac{\partial g_1}{\partial x_2}\right|_{(x_1,x_2)} \\ \left.\frac{\partial g_2}{\partial x_1}\right|_{(x_1,x_2)} & \left.\frac{\partial g_2}{\partial x_2}\right|_{(x_1,x_2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - h \begin{bmatrix} -2x_1 & 1 \\ 0 & -100x_2 \end{bmatrix}$$

Schémas imbriqués

Euler implicite

On fournit $\mathbf{U}_0$

Pour chaque i, on calcule $\mathbf{x} = \mathbf{U}_{i+1}$ à partir de $\mathbf{U}_i$ en résolvant le problème

$$\underbrace{\mathbf{U}_{i+1} - \mathbf{U}_i - hf(\mathbf{U}_{i+1}, X_{i+1})}_{\mathbf{g}(\mathbf{x})} = 0$$

par la méthode de Newton-Raphson en partant de $\mathbf{U}_i$

Newton-Raphson

On fournit $\mathbf{x}_0 = \mathbf{U}_i$

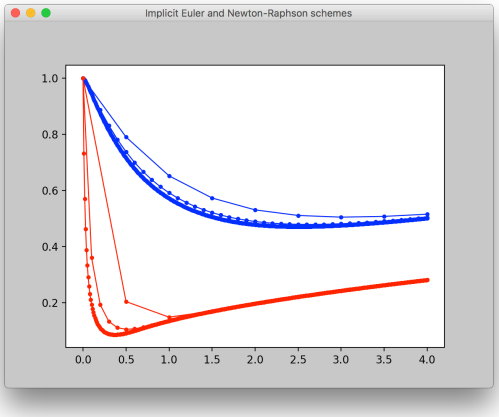
Tant que $\Delta\mathbf{x} > \epsilon$, on calcule $\mathbf{x}_{j+1}$ à partir de $\mathbf{x}_j$ avec

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}_j)\overbrace{(\mathbf{x}_{j+1} - \mathbf{x}_j)}^{\Delta\mathbf{x}} = -\mathbf{g}(\mathbf{x}_j)$$

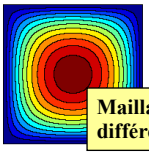
$$\mathbf{x}_{j+1} = \mathbf{x}_j + \Delta\mathbf{x}$$

Si on converge, la solution $\mathbf{U}_{i+1}$ est le dernier $\mathbf{x}_{j+1}$ calculé

Que fournit l'utilisation conjointe de méthodes d'Euler implicite et de Newton-Raphson ?

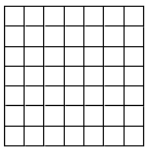


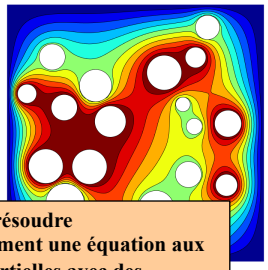
A quoi servent les méthodes numériques ?



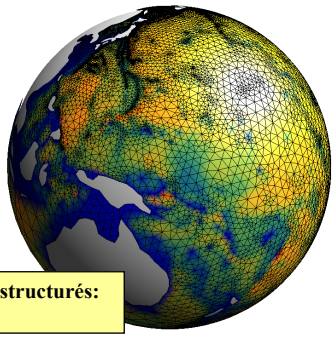
LEPL1104

Maillages structurés: différences finies





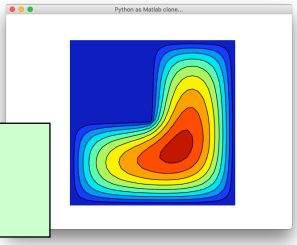
Comment résoudre numériquement une équation aux dérivées partielles avec des conditions aux limites ?



LEPL1110

Maillages non-structurés: éléments finis

Plan du cours de méthodes numériques



Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non-linéaires ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Trois problèmes aux conditions aux limites

Fonction inconnue

Elliptique

Trouver $u(x,y)$ tel que

$$\begin{aligned} \nabla^2 u(x,y) + f(x,y) &= 0, && (x,y) \in \Omega, \\ u(x,y) &= \overline{u}(x,y), && (x,y) \in \partial\Omega, \end{aligned}$$

Maillages structurés: différences finies

Parabolique

Trouver $u(x,y,t)$ tel que

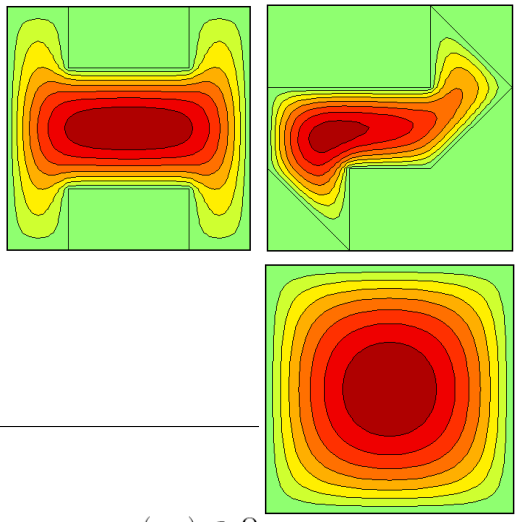
$$\begin{aligned} \frac{\partial u}{\partial t}(x,y,t) &= \nabla^2 u(x,y,t) + f(x,y,t), && (x,y) \in \Omega, \\ u(x,y,t) &= \overline{u}(x,y,t), && (x,y) \in \partial\Omega, \\ u(x,y,0) &= u_0(x,y), && (x,y) \in \Omega, \end{aligned}$$

Hyperbolique

Trouver $u(x,y,t)$ tel que

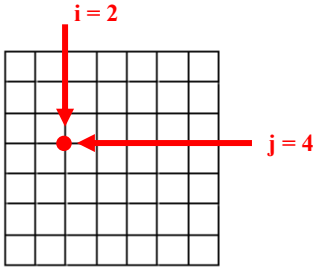
$$\begin{aligned} \frac{\partial^2 u}{\partial t^2}(x,y,t) &= \nabla^2 u(x,y,t) + f(x,y,t), && (x,y) \in \Omega, \\ u(x,y,t) &= \overline{u}(x,y,t), && (x,y) \in \partial\Omega, \\ u(x,y,0) &= u_0(x,y), \quad \frac{\partial u}{\partial t}(x,y,0) = 0, && (x,y) \in \Omega, \end{aligned}$$

Problème elliptique…



Trouver $u(x,y)$ tel que

$$\begin{aligned} \nabla^2 u(x,y) + f(x,y) &= 0, && (x,y) \in \Omega, \\ u(x,y) &= \overline{u}(x,y), && (x,y) \in \partial\Omega, \end{aligned}$$



Grille / Maillage

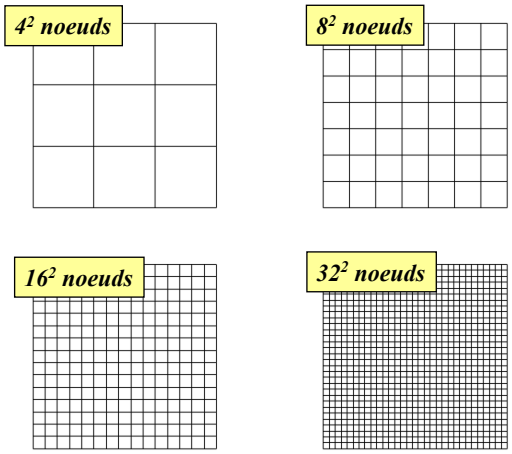
$$X_i = a + ih, \quad i = 0, \ldots, m$$

$$h = (b - a)/(m)$$

$$\mathbf{X}_{ij} = (X_i, Y_j) = (ih, jh)$$

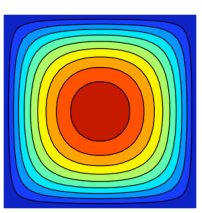
$$U_{ij} = u^h(\mathbf{X}_{ij}) \approx u(\mathbf{X}_{ij})$$

Méthode des différences finies



Exemple

$$\begin{aligned} \nabla^2 u(x,y) + 1 &= 0, \quad (x,y) \in \Omega, \\ u(x,y) &= 0, \quad (x,y) \in \partial\Omega, \end{aligned}$$



$$u(x,y) = \sum_{i,j \ impairs} C_{ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right)$$

Comment trouver C_{ij} ?

$$\nabla^2 u(x,y) + 1 = 0,$$

$$\sum_{i,j \text{ impairs}} \frac{-\pi^2(i^2+j^2)}{4} C_{ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right) + 1 = 0,$$

En vertu de l'orthogonalité des sinus,

$$\frac{\pi^2(i^2+j^2)}{4} C_{ij} - \underbrace{\int_\Omega \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right) d\Omega}_{16/(ij\pi^2)} = 0,$$

$$u(x,y) = \sum_{i,j \ impairs} \frac{64}{\pi^4(i^2+j^2)ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right)$$

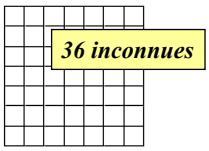
Différences finies

$$\left(\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h^2}\right) + 1 = 0$$

$$\frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2} + 1 = 0$$

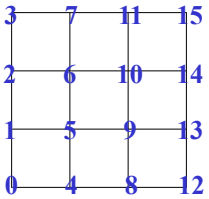
(n-2)(n-2) équations linéaires
(n-2)(n-2) inconnues

Il suffit donc de résoudre un grand système linéaire



Programme Python….

```
def poissonSolve(nx,ny):
  n = nx*ny; h = 2/(ny-1)
  A = eye(n); B = zeros(n)
  for i in range(1,nx-1):
    for j in range(1,ny-1):
      index = i + j*nx
      A[index,index]     =  4.0
      A[index,index-1]   = -1.0
      A[index,index+1]   = -1.0
      A[index,index-nx] = -1.0
      A[index,index+nx] = -1.0
      B[index] = 1
  return solve(A,B).reshape(ny,nx)
```



$$\frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2} + 1 = 0$$

Comment résoudre le système discret avec python...

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

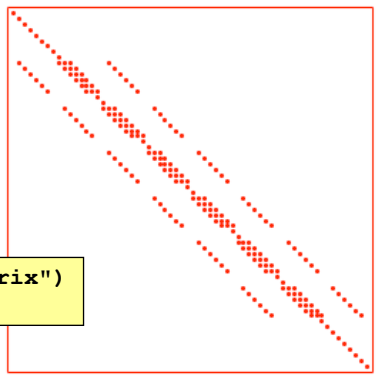
```
x = solve(A,b)
```

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 solve function

x = solve(A,b)

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

Les différences finies produisent une matrice creuse…

```
plt.figure("Sparcity of the matrix")
plt.spy(A,marker='o',color='r)
```



Utiliser scipy.sparse !

```
from scipy.sparse import dok_matrix
from scipy.sparse.linalg import spsolve

n = nx*ny; h = 2/(ny-1)
A = dok_matrix((n,n),dtype=float32);
B = zeros(n)
for i in range(n):
  A[i,i] = 1.0
for i in range(1,nx-1):
  for j in range(1,ny-1):
    index = i + j*nx
    A[index,index]    =  4.0
    A[index,index-1]  = -1.0
    A[index,index+1]  = -1.0
    A[index,index-nx] = -1.0
    A[index,index+nx] = -1.0
    B[index] = 1
U = spsolve(A/(h*h).tocsr(),B).reshape(ny,nx)
```

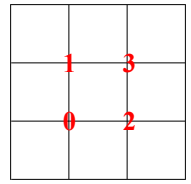
Pour les matrices de grande taille c'est plus rapide !

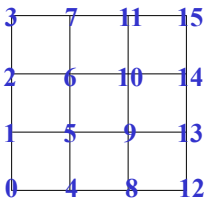
```
bash-3.2$ python poissonSimple.py
 === Considering nx=ny=150
 === Full solver   : elapsed time is 68.746733 seconds.
 === Sparse solver : elapsed time is 1.635676 seconds.
```

https://docs.scipy.org/doc/scipy-0.9.0/reference/sparse.html

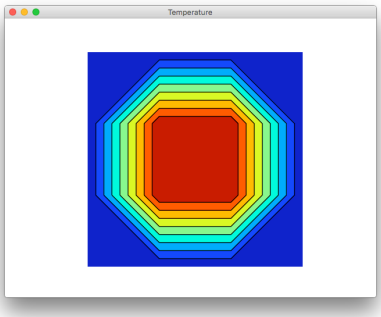
Petits problèmes de numérotation…

```
n = nx*ny; h = 2/(ny-1)
A = zeros((n,n))
B = ones((nx-2)*(ny-2))
map = zeros((nx-2)*(ny-2),dtype=int)
for i in range(1,nx-1):
  for j in range(1,ny-1):
    index = i + j*nx
    map[i-1 + (j-1)*(nx-2)] = index
    A[index,index]     =  4.0
    A[index,index-1]   = -1.0
    A[index,index+1]   = -1.0
    A[index,index-nx] = -1.0
    A[index,index+nx] = -1.0
A = A / (h*h)
V = solve(A[map,:][:,map],B)
```

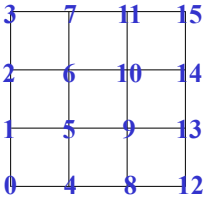


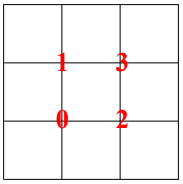



```
U = zeros((ny,nx))
U[1:ny-1,1:nx-1] = V.reshape(ny-2,nx-2)

X,Y = meshgrid(linspace(-1,1,nx),linspace(-1,1,ny))
myColorMap = matplotlib.cm.jet
plt.contour(X,Y,U,10,cmap=myColorMap)
plt.contour(X,Y,U,10,colors='k',linewidths=1)
plt.axis("equal"); plt.axis("off")
```

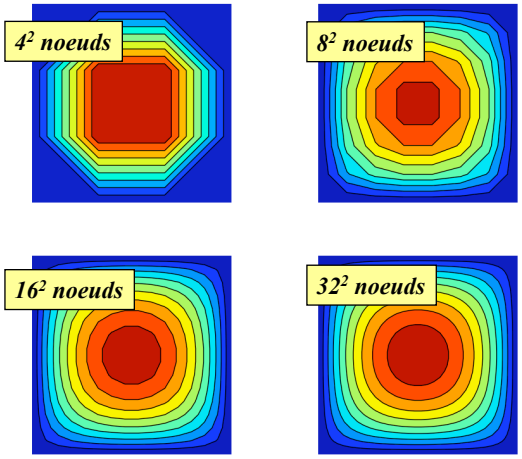


Et faire un joli plot

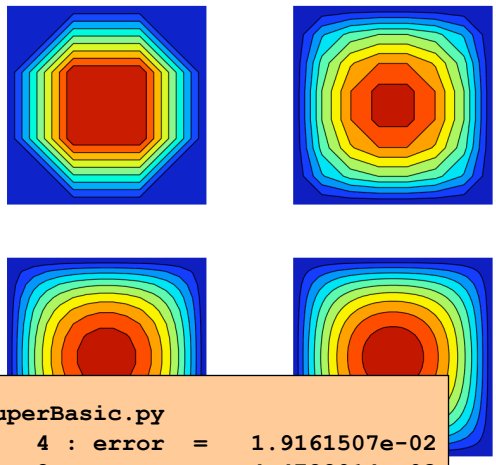




Est-ce que cela converge ?



Quelle est la précision ?



```
bash-3.2$ python poissonSuperBasic.py
 === Discretization nx =    4 : error  =   1.9161507e-02
 === Discretization nx =    8 : error  =   4.4728014e-03
 === Discretization nx =   16 : error  =   1.0188850e-03
 === Discretization nx =   32 : error  =   2.4094496e-04

 ===== Estimated order of convergence  =   2.1044545e+00
```

Théoriquement…

Théorème 6.1. *Soit $u(x, y)$ la solution exacte d'un problème de Poisson aux conditions aux limites sur un domaine $\Omega \subset \mathbb{R}^2$.*
Si la fonction u et toutes ses dérivées partielles jusqu'au quatrième ordre sont continues sur le domaine fermé $\overline{\Omega}$, alors il existe une constante positive telle que :

$$\max |u(X_i, Y_j) - \underbrace{u^h(Xi, Y_j)}_{U_{ij}}| \leq C\ M\ h^2$$

où M est la valeur maximale atteinte par une des dérivées quatrièmes de u sur $\overline{\Omega}$ et u^h est la solution discrète obtenue au moyen d'un schéma à 5 points basé sur des différences finies centrées du second ordre.

Ce résultat n'est pas évident a priori…
Démonstration :
Isaacson and Keller, Analysis of Numerical Methods (1966)

Plus compliqué :-)

```
m = 5
map = numgrid(2*m+1)
print(map)
```

```
[[ 0  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8  9  0]
 [ 0 10 11 12 13 14 15 16 17 18  0]
 [ 0 19 20 21 22 23 24 25 26 27  0]
 [ 0 28 29 30 31 32 33 34 35 36  0]
 [ 0  0  0  0  0  0 37 38 39 40  0]
 [ 0  0  0  0  0  0 41 42 43 44  0]
 [ 0  0  0  0  0  0 45 46 47 48  0]
 [ 0  0  0  0  0  0 49 50 51 52  0]
 [ 0  0  0  0  0  0 53 54 55 56  0]
 [ 0  0  0  0  0  0  0  0  0  0  0]]
```

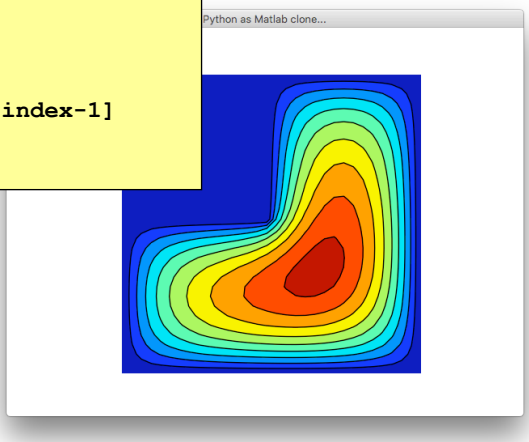
Et pourtant si simple…

```
m = 3;
map = numgrid(2*m+1)
A = delsq(map); print(A.toarray())
```

```
[[ 4. -1.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [-1.  4. -1.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  4. -1.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  4. -1.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  4.  0.  0.  0.  0. -1.  0.  0.  0.  0.  0.]
 [-1.  0.  0.  0.  0.  4. -1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0. -1.  4. -1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0.  0.  0. -1.  4. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.  0.  0. -1.  4. -1. -1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  0.  0.  0. -1.  4.  0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  4. -1. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -1. -1.  4.  0. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  4. -1. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1. -1.  4.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  4.]]
```

Et finalement…

```
map = numgrid(32)
index = map[map>0]

A = delsq(map)
B = ones(size(index))
U = zeros(shape(map))
U[map>0] = spsolve(A,B)[index-1]

plt.contourf(U,10)
```



Malheureusement,
les fonctions numgrid et delsq
de MATLAB ne font pas encore
partie de numpy et de scipy !

Problème parabolique…

Trouver $u(x,y,t)$ tel que

$$\begin{aligned} \frac{\partial u}{\partial t}(x,y,t) &= \boldsymbol{\nabla}^2 u(x,y,t) + f(x,y,t), && (x,y) \in \Omega, \\ u(x,y,t) &= \overline{u}(x,y,t), && (x,y) \in \partial\Omega, \\ u(x,y,0) &= u_0(x,y), && (x,y) \in \Omega, \end{aligned}$$

Exemple

$$\begin{cases} \rho c \dfrac{\partial T}{\partial t}(x,t) &= k \dfrac{\partial^2 T}{\partial x^2}(x,t), \\ T(0,t) &= T_{in}, \\ -k \dfrac{\partial T}{\partial y}(L,t) &= h\Big(T(L,t) - T_{out}\Big), \\ T(x,0) &= T_{in}. \end{cases}$$

Différences finies (espace)
Euler explicite (temps)

$$\rho c\left(\frac{T_i^{n+1}-T_i^n}{\Delta t}\right) = k\left(\frac{T_{i+1}^n-2T_i^n+T_{i-1}^n}{(\Delta x)^2}\right)$$

$$\downarrow \quad \text{En définissant } \beta = \frac{k\Delta t}{\rho c(\Delta x)^2},$$

$$T_i^{n+1} = T_i^n + \beta\left(T_{i+1}^n + T_{i-1}^n - 2T_i^n\right)$$

C'est une itération pour un vecteur qui doit converger vers la solution de régime
C'est quelque chose qu'on a déjà rencontré…

Oublions toutes ces températures et reprenons notre notation générique…

En définissant $\alpha = k/(\rho c)$

$$\left(\frac{U_i^{n+1}-U_i^n}{\Delta t}\right) = \alpha\left(\frac{U_{i+1}^n-2U_i^n+U_{i-1}^n}{(\Delta x)^2}\right)$$

$$\downarrow \quad \text{En définissant } \beta = \frac{\alpha\Delta t}{(\Delta x)^2},$$

$$U_i^{n+1} = U_i^n + \beta\left(U_{i+1}^n + U_{i-1}^n - 2U_i^n\right)$$

En fait, on intègre un système linéaire…

$$\mathbf{u}' = \frac{\alpha}{(\Delta x)^2}\mathbf{A}\mathbf{u}$$

$$\mathbf{u}_n = \begin{bmatrix} U_1^n \\ U_2^n \\ U_3^n \\ U_4^n \\ U_5^n \\ \\ \\ U_{n_x}^n \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} -2 & 1 & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & 1 & -2 & 1 & & & \\ & & & 1 & -2 & 1 & & \\ & & & & & & 1 & -2 \end{bmatrix}$$

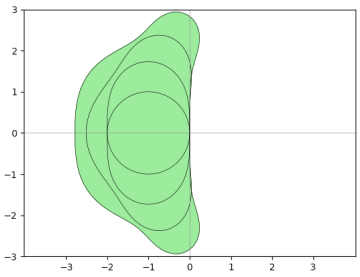
Stabilité des systèmes

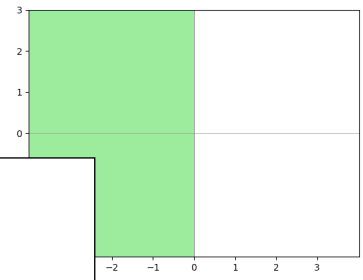
$$\mathbf{u}' = \mathbf{A}\mathbf{u}$$

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$$

$$\mathbf{v}(x) = \mathbf{P}^{-1}\mathbf{u}(x)$$

$$\mathbf{v}' = \mathbf{D}\mathbf{v}$$





Le problème différentiel initial est équivalent à n équations scalaires

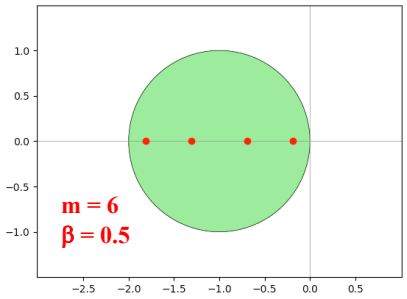
$$v_i'(x) = \lambda_i v_i(x), \qquad i = 1, \ldots, n$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \beta \mathbf{A}\mathbf{u}_n$$

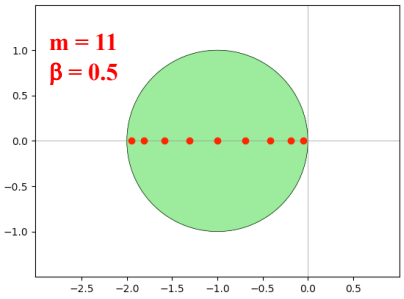
Euler explicite

$$|1 + \beta\lambda_i| < 1$$

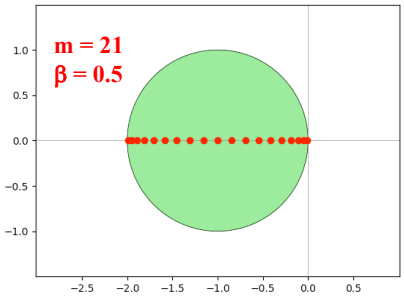
$\Delta x = 0.2, \Delta t = 0.02$



$\Delta x = 0.1, \Delta t = 0.005$



En raffinant le maillage…



$\Delta x = 0.05, \Delta t = 0.00125$

Adaptons…

$$\mathbf{u}_{n+1} = \underbrace{\left(\mathbf{I} + \beta\mathbf{A}\right)}_{\mathbf{M}} \mathbf{u}_n$$

Condition pour qu'une méthode itérative converge ?

$$\underbrace{(\mathbf{x}_{i+1} - \mathbf{x})}_{\mathbf{e}_{i+1}} = \mathbf{M}\mathbf{x}_i + \mathbf{c} - \mathbf{M}\mathbf{x} - \mathbf{c} \qquad \mathbf{x}_{i+1} = \mathbf{M}\mathbf{x}_i + \mathbf{c}$$

$$= \mathbf{M}\,(\mathbf{x}_i - \mathbf{x})$$

En procédant de la même manière pour chaque étape,

$$= \mathbf{M}^{i+1} \underbrace{(\mathbf{x}_0 - \mathbf{x})}_{\mathbf{e}_0}$$

Il faut que…

$$\lim_{i \to \infty} \mathbf{M}^i \mathbf{e}_0 = 0$$

$$\mathbf{u}_n = \begin{bmatrix} U_1^n \\ U_2^n \\ U_3^n \\ U_4^n \\ U_5^n \\ \\ \\ U_{n_x}^n \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} -2 & 1 & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & 1 & -2 & 1 & & & \\ & & & 1 & -2 & 1 & & \\ & & & & & & & \\ & & & & & & 1 & -2 \end{bmatrix}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \beta\mathbf{A}\mathbf{u}_n$$

$$|1 + \beta\lambda_i| < 1$$

Et toujours, les valeurs propres…

Ecrivons l'erreur comme une combili de vecteurs propres…

$$\mathbf{e}_0 = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j$$

Puisque $\mathbf{M}\,\mathbf{v}_i = \lambda_i \mathbf{v}_i$,

$$\mathbf{e}_i = \sum_{j=1}^{n} \alpha_j \lambda_j^i \mathbf{v}_j$$

Pour obtenir…

$$\lim_{i \to \infty} \mathbf{M}^i \mathbf{e}_0 = 0$$

… on doit exiger que le rayon spectral de la matrice M soit inférieur à l'unité

$$|\lambda_i| < 1 \qquad \forall\, i$$

Mais, il faut calculer les valeurs propres d'un opérateur laplacien discret quelconque….

$$\beta = \frac{\alpha \Delta t}{(\Delta x)^2} \leq \frac{1}{2}$$

Courant, Friedrichs et Lewy (1928)

Eux, ils ne disposaient pas de Python….

Analyse de stabilité

Amplitude quelconque de la perturbation

Nombre imaginaire

$$U_i^n = U^n e^{ikX_i}$$

Indice spatial

k quelconque

Indice spatial

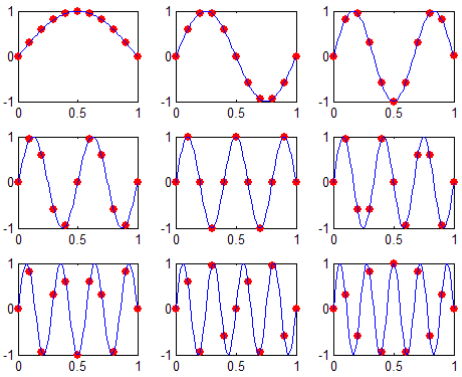
Considérons une perturbation quelconque de la forme suivante et analysons son évolution….
On souhaite que son amplitude diminue.

Quelques k bien choisis…

m = 11
β = 0.5

$$U_i^n = U^n \sin(\frac{\hat{k}\pi X_i}{L})$$

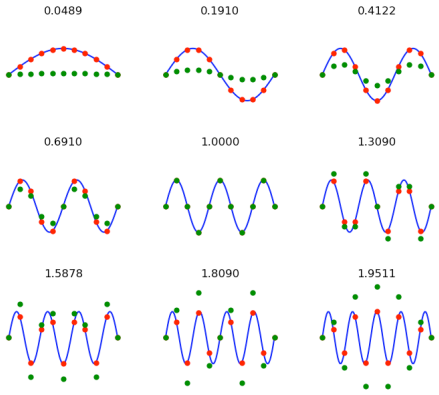
$\Delta x = 0.1$



Des perturbations bien particulières et propres…

m = 11
β = 0.5

$$U_i^n = U^n \sin(\frac{\hat{k}\pi X_i}{L})$$



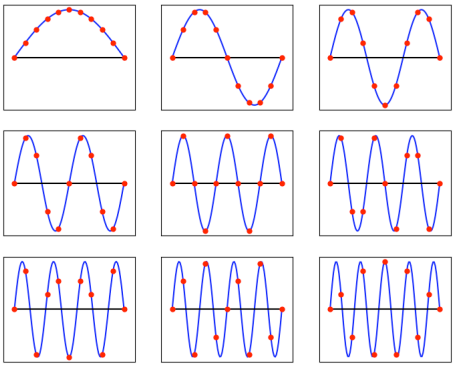
u

$-\beta \mathbf{A}\mathbf{u}$

C'est quoi ces perturbations ?

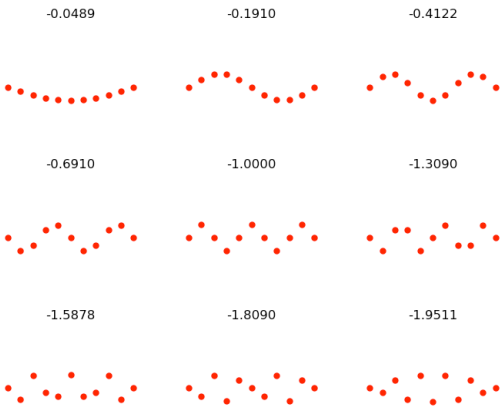
N'importe quelle perturbation peut être écrite comme une combinaison linéaire de ces vecteurs propres de notre opérateur….

m = 11
β = 0.5



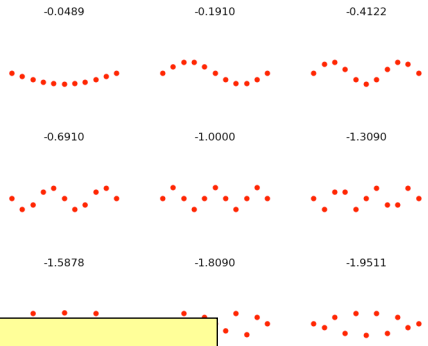
Dimension de l'espace discret = 9
Nombre de vecteurs de base = 9

Calculons les valeurs et les vecteurs propres…

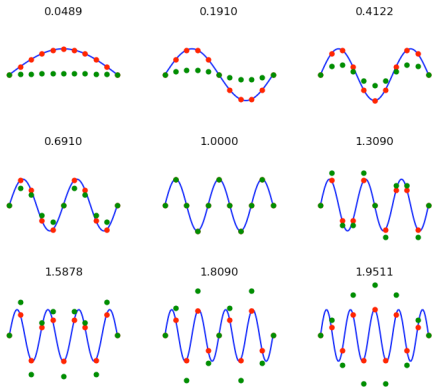



```
m = 11; dx = 1.0/(m-1); beta = (dx*dx)/2
e = array([0,*ones(m-2),0])
D = spdiags([e,-2*e,e],[-1,0,1],m,m)
D = D.T/(dx*dx)
lam,eigen = eig(beta * D.toarray()[1:-1,1:-1])
```

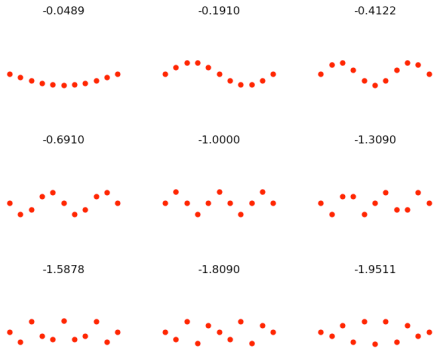
Trier et dessiner les vecteurs propres




```
order = argsort(-lam.real)
X = linspace(0,1,m); V = zeros(m)
for i in range(m-2):
  plt.subplot(msqrt,msqrt,i+1)
  V[1:-1] = eigen[:,order[i]]
  plt.plot(X,V,'.r',markersize=10)
  plt.title("%6.4f" % lam[order[i]].real)
  plt.xlim((-0.1,1.1))
  plt.ylim((-2.1,2.1))
  plt.axis('off')
```



$$U_i^n = U^n \sin(\frac{\hat{k}\pi X_i}{L})$$



Et c'est vraiment cela ?

Propagation des erreurs

$$
\begin{aligned}
U_i^{n+1} &= U_i^n + \beta\Big(U_{i+1}^n + U_{i-1}^n - 2U_i^n\Big) \\
&= U_i^n\Big(1 + \beta\Big(e^{ik\Delta x} + e^{-ik\Delta x} - 2\Big)\Big) \\
&= U_i^n\Big(1 + \beta\Big(2\cos(k\Delta x) - 2\Big)\Big)
\end{aligned}
$$

Pour que les erreurs ne s'amplifient pas, il faut que la valeur absolue de ce facteur soit inférieure à l'unité pour tout k…

$$U = \Big(1 + \beta\Big(2\cos(k\Delta x) - 2\Big)\Big)$$

Un peu d'algèbre …

$$|1 + \beta\Big(2\cos(k\Delta x) - 2\Big)| \leq 1$$

En prenant le cas le plus défavorable où $k\Delta x = \pi$,

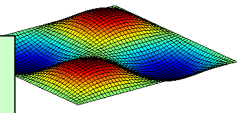
$$
\begin{aligned}
-1 &\leq 1 - 4\beta \\
2\beta &\leq 1
\end{aligned}
$$

$$\beta = \frac{\alpha\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$$

Condition de stabilité très pénalisante sur le pas de temps….

Courant, Friedrichs et Lewy (1928)

Plan du cours de méthodes numériques



Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non-linéaires ?

Et les méthodes itératives ?

Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Comment résoudre numériquement une équation aux dérivées partielles ?

Exemple : corde vibrante

Conditions intiales

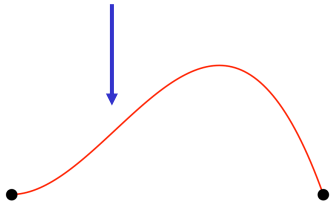
$$u(x,0) = u_0 \left(1 - \frac{x}{L}\right)\left(\frac{x}{L}\right)^2$$

$$\frac{\partial u}{\partial t}(x,0) = 0$$

Tension présente dans la corde [N]

$$\rho \frac{\partial^2 u}{\partial t^2} = T_0 \frac{\partial^2 u}{\partial x^2},$$

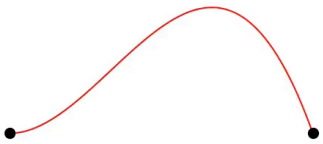
Masse par unité de longueur [kg/m]



$$u(0,t) = 0 \qquad u(L,t) = 0$$

Conditions aux limites

```
from matplotlib import animation

def frame(i):
  L  = 1; u0 = 1; c  = 1
  t = i*L/(c*10); x = linspace(0,L,100)
  plt.clf(); plt.axis("off")
  plt.plot(x,waveAnalytic(x,t,L,c,u0),"-r")

movie = animation.FuncAnimation(plt.figure(),frame,200)
movie.save('wave.mp4',fps=30)
```



Faire un film avec matplolib

Solution analytique

$$u(x,t) = \sum_{n=1}^{\infty} C_n \sin\left(\frac{n\pi x}{L}\right) \cos\left(\frac{n\pi ct}{L}\right)$$

Conditions intiales

$$u(x,0) = u_0 \left(1 - \frac{x}{L}\right)\left(\frac{x}{L}\right)^2$$

$$\frac{\partial u}{\partial t}(x,0) = 0$$

$$\rho \frac{\partial^2 u}{\partial t^2} = T_0 \frac{\partial^2 u}{\partial x^2},$$

$$u(0,t) = 0 \qquad u(L,t) = 0$$

Conditions aux limites

Comment satisfaire la condition initiale ?

$$u(x,0) = u_0\left(\frac{x^2}{L^2} - \frac{x^3}{L^3}\right),$$

$$\sum_{n=1}^{\infty} C_n \sin\left(\frac{n\pi x}{L}\right) = u_0\left(\frac{x^2}{L^2} - \frac{x^3}{L^3}\right),$$

En vertu de l'orthogonalité des sinus,

$$\frac{L}{2} C_n = u_0 \int_0^L \left(\frac{x^2}{L^2} - \frac{x^3}{L^3}\right) \sin\left(\frac{n\pi x}{L}\right) dx,$$

Un peu d'algèbre

$$C_n = 2u_0 \int_0^1 (t^2 - t^3) \sin(n\pi t)\, dt$$

$$= 2u_0 \left[\left(\frac{2-(n\pi)^2 t^2}{(n\pi)^3} - \frac{6t-(n\pi)^2 t^3}{(n\pi)^3}\right)\cos(n\pi t)\right]_0^1$$

$$= 2u_0 \left(\frac{2-(n\pi)^2-6+(n\pi)^2}{(n\pi)^3}(-1)^n - \frac{2}{(n\pi)^3}\right) = \frac{4u_0}{n^3\pi^3}\left(2(-1)^{n+1}-1\right)$$

```
def waveAnalyticBasic(x,t,L,c,u0):
  u = zeros(size(x))
  for n in range(1,201):
    u = u + ( sin(n*pi * x/L)
            * cos(n*pi * c*t/L)
            * (2*(-1)**(n+1) - 1)/(n**3) )
  u = u * 4.0 * u0/(pi**3)
  return u
```

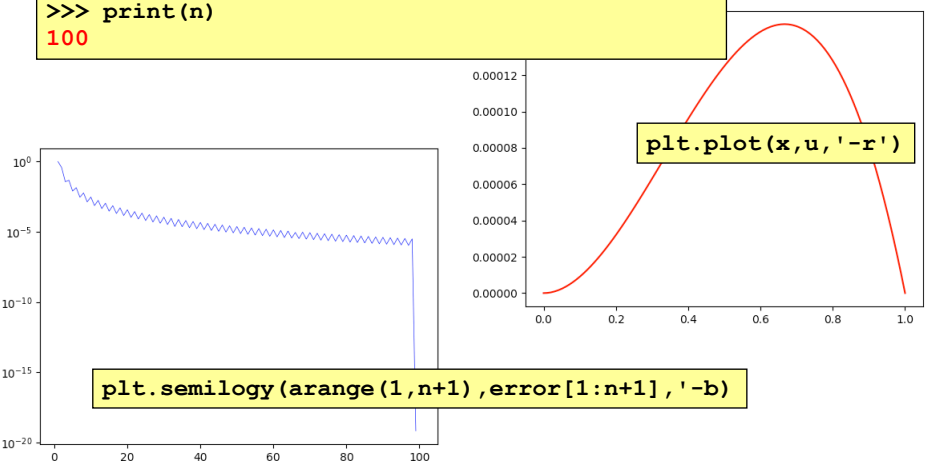
10, 50, 200 ou 1000 termes…
Comment choisir ?

10, 20, 50 ou 100 termes…
Comment choisir ?

```
def waveAnalyticStupid(x,t,L,c,u0):
  tol = 1e-8; nmax = 2000; n = 1; errorloc = 2*tol
  u = zeros(size(x)); delta = zeros(size(x))
  error = zeros(nmax+1)
  while errorloc > tol and n < nmax:
    delta = ( sin(n*pi * x/L)
            * cos(n*pi * c*t/L)
            * (2*(-1)**(n+1) - 1)/(n**3) )
    u = u + delta
    errorloc  = max(abs(delta))
    error[n] = errorloc
    n = n+1
  u = u * 4.0 * u0/(pi**3)
  if (n == nmax) :
    print("Error, not converged yek, yek :-(")
  return u,error,n
```

Et boum !

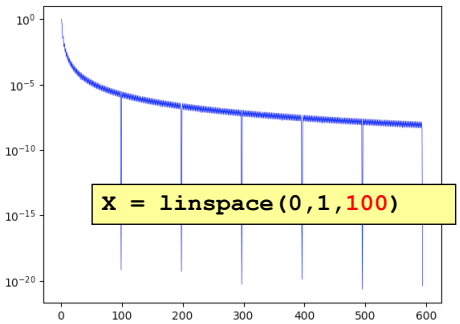
```
>>> x = linspace(0,1,100)
>>> u,error,n = waveAnalytic(x,0,L,c,u0)
>>> print(n)
100
```

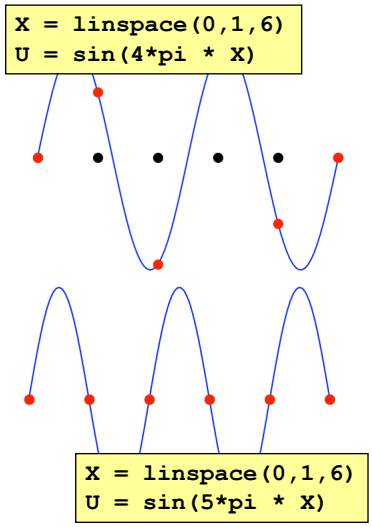


Une meilleure implémentation !

```
def waveAnalytic(x,t,L,c,u0):
  tol = 1e-8; nmax = 2000; n = 1; errorloc = 2*tol
  u = zeros(size(x)); delta = zeros(size(x))
  error = zeros(nmax+1)
  error[0] = errorloc
  while errorloc > tol and n < nmax:
    delta = ( sin(n*pi * x/L)
            * cos(n*pi * c*t/L)
            * (2*(-1)**(n+1) - 1)/(n**3) )
    u = u + delta
    error[n] = max(abs(delta))
    errorloc = max(error[n],error[n-1])
    n = n+1
  u = u * 4.0 * u0/(pi**3)
  if (n == nmax) :
    print("Error, not converged yek, yek :-(")
  return u,error,n
```

Termes de la série





$$u(x,t) = \frac{4u_0}{\pi^3}\sum_{n=1}^{\infty}\frac{1}{n^3}\left(2(-1)^{n+1}-1\right)\sin\left(\frac{n\pi x}{L}\right)\cos\left(\frac{n\pi ct}{L}\right)$$

Différences finies spatiales

Equation aux dérivées partielles du second ordre

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

$$U_i(t) = u^h(X_i, t) \approx u(X_i, t) \qquad i = 1, \ldots, m$$

$$\frac{d^2 U_i}{dt^2} = c^2 \frac{U_{i+1} - 2U_i + U_{i-1}}{(\Delta x)^2}$$

Système de m équations différentielles ordinaires du second ordre

Deux options possibles pour la discrétisation temporelle

$$\frac{d^2 U_i}{dt^2} = c^2 \frac{U_{i+1} - 2U_i + U_{i-1}}{(\Delta x)^2}$$

Système de m équations différentielles ordinaires du second ordre

$$\begin{cases} \dfrac{dU_i}{dt} = V_i \\ \dfrac{dV_i}{dt} = c^2 \dfrac{U_{i+1} - 2U_i + U_{i-1}}{(\Delta x)^2} \end{cases}$$

Système de 2m équations différentielles ordinaires du premier ordre

$$\begin{cases} \dfrac{dU_i}{dt} &= V_i \\ \dfrac{dV_i}{dt} &= c^2 \dfrac{U_{i+1} - 2U_i + U_{i-1}}{(\Delta x)^2} \end{cases}$$

2m équadiffs !

$$\frac{dY_i}{dt} = \sum_{j=1}^{2m} A_{ij} Y_j$$

$$Y_i = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{bmatrix}, \qquad A_{ij} = \left[\begin{array}{ccccc|ccccc} & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \\ \hline 0 & & & & & & & & & \\ a & -2a & a & & & & & & & \\ & a & -2a & a & & & & & & \\ & & a & -2a & a & & & & & \\ & & & & 0 & & & & & \end{array}\right] \qquad a = \frac{c^2}{(\Delta x)^2}$$

Comment construire cette matrice ?

$$A_{ij} = \left[\begin{array}{ccccc|ccccc} 0 & & & & & 1 & & & & \\ a & -2a & a & & & & 1 & & & \\ & a & -2a & a & & & & 1 & & \\ & & a & -2a & a & & & & 1 & \\ & & & & 0 & & & & & 1 \end{array}\right] \qquad a = \frac{c^2}{(\Delta x)^2}$$

```
m = 21; dx = L/(m-1)

e = array([0,*ones(m-2),0])
D = spdiags([e,-2*e,e],[-1,0,1],m,m)
Dt = (D.T)*(c*c)/(dx*dx)
```

Equations différentielles ordinaires …. avec RK45 :-)

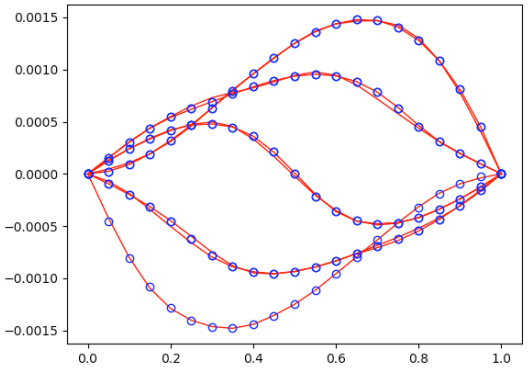
$$\frac{dY_i}{dt} = \sum_{j=1}^{2m} A_{ij} Y_j$$

```
x = linspace(0,L,m)
y0 = zeros(2*m);
y0[0:m] = u0 * ((x/L)**2)*(1 - x/L)
sol = solve_ivp(dfdt,[0,Lt],y0,method="RK45",
                                rtol=1e-6,atol=1e-9)
```

```
def dfdt(t,y):
  u = y[:m]; v = y[m:]
  dudt = v;  dvdt = Dt * u
  return [*dudt,*dvdt]
```

Résultats par RK45 :-)

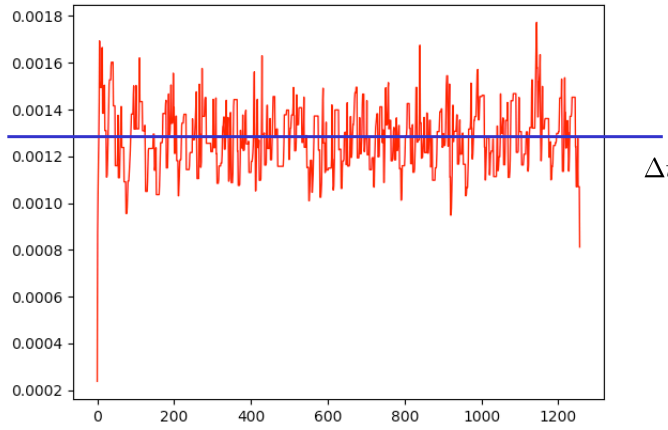
RK45 : 21 valeurs nodales (m=20)



Solution analytique

Résultat sur une période

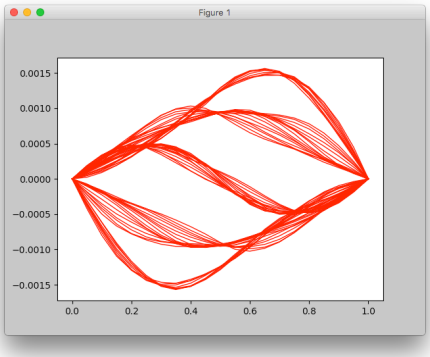
Evolution du pas de temps



$\Delta t \approx 10^{-3}$

Comment obtenir une courbe à un instant précis ?

La courbe à un instant fixé est obtenue par une interpolation linéaire entre les courbes des deux pas de temps les plus proches…




```
nP = 8
t = sol.t; tplot = linspace(0,Lt,8*nP+1)
y = sol.y; yplot = interp1d(t,y)(tplot)
plt.plot(x,yplot[:m,:],'-r',linewidth=1)
```

Les 65 courbes sont dessinées en une seule instruction…

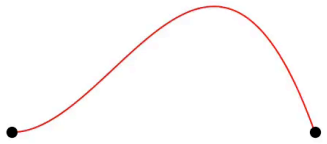
Energie cinétique

$$E_c(t) = \frac{\rho}{2}\int_0^L \left(\frac{\partial u}{\partial t}(x,t)\right)^2 dx$$

$$E_p(t) = \frac{T_0}{2}\int_0^L \left(\frac{\partial u}{\partial x}(x,t)\right)^2 dx$$

Energie potentielle

Un peu de physique



L'énergie totale est conservée…

$$\overbrace{E_c(t)+E_p(t)}^{E} = \frac{\rho}{2}\int_0^L \left(\frac{\partial u}{\partial t}\right)^2 dx + \frac{\rho c^2}{2}\int_0^L \left(\frac{\partial u}{\partial x}\right)^2 dx$$

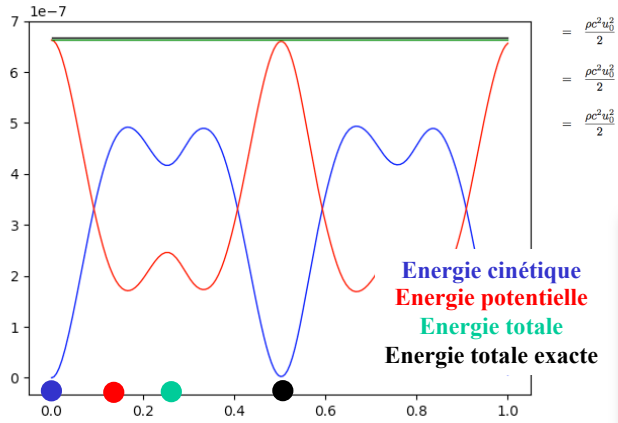
$$\frac{dE}{dt} = \rho\int_0^L \left(\frac{\partial^2 u}{\partial t^2}\frac{\partial u}{\partial t} + c^2\frac{\partial u}{\partial x}\frac{\partial^2 u}{\partial x \partial t}\right) dx$$

$$= \rho\int_0^L \left(c^2\frac{\partial^2 u}{\partial x^2}\frac{\partial u}{\partial t} + c^2\frac{\partial u}{\partial x}\frac{\partial^2 u}{\partial x \partial t}\right) dx$$

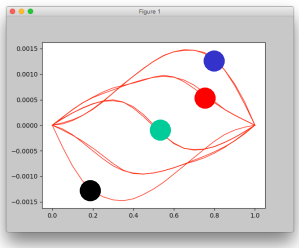
$$= \rho c^2\int_0^L \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\frac{\partial u}{\partial t}\right) dx = \rho c^2\left[\frac{\partial u}{\partial x}\frac{\partial u}{\partial t}\right]_0^L$$

Numériquement…

$$
\begin{aligned}
E &= \frac{\rho c^2}{2}\int_0^L \left(\frac{\partial u}{\partial x}(x,0)\right)^2 dx \\
&= \frac{\rho c^2 u_0^2}{2}\int_0^L \left(\frac{2x}{L^2}-\frac{3x^2}{L^3}\right)^2 dx \\
&= \frac{\rho c^2 u_0^2}{2}\int_0^L \left(\frac{4x^2}{L^4}-\frac{12x^3}{L^5}+\frac{9x^4}{L^6}\right) dx \\
&= \frac{\rho c^2 u_0^2}{2}\left[\frac{4x^3}{3L^4}-\frac{12x^4}{4L^5}+\frac{9x^5}{5L^6}\right]_0^L \\
&= \frac{\rho c^2 u_0^2}{2}\left(\frac{4}{3L}-\frac{3}{L}+\frac{9}{5L}\right) = \frac{\rho c^2 u_0^2}{2L}\left(\frac{20-45+27}{15}\right) = \frac{\rho c^2 u_0^2}{15L}
\end{aligned}
$$



Résultat sur <u>une</u> période



Deux options possibles pour la discrétisation temporelle

$$\frac{d^2U_i}{dt^2} = c^2\frac{U_{i+1}-2U_i+U_{i-1}}{(\Delta x)^2}$$

Système de m équations différentielles ordinaires du second ordre

$$\begin{cases} \dfrac{dU_i}{dt} = V_i \\ \dfrac{dV_i}{dt} = c^2\dfrac{U_{i+1}-2U_i+U_{i-1}}{(\Delta x)^2} \end{cases}$$

Système de 2m équations différentielles ordinaires du premier ordre

Différences finies centrées…

Système de m équations différentielles ordinaires du second ordre

$$\frac{d^2U_i}{dt^2} = c^2\frac{U_{i+1} - 2U_i + U_{i-1}}{(\Delta x)^2}$$

$$\downarrow$$

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{(\Delta t)^2} = c^2\frac{U_{i+1}^n - 2U_i^n + U_i^n}{(\Delta x)^2}$$

Système de m équations aux récurrences à deux termes

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{(\Delta t)^2} = c^2\frac{U_{i+1}^n - 2U_i^n + U_i^n}{(\Delta x)^2}$$

$$\downarrow$$

$$U_i^{n+1} = 2U_i^n + \beta^2\left(U_{i+1}^n - 2U_i^n + U_{i-1}^n\right) - U_i^{n-1}$$

$$\beta = \frac{c\Delta t}{\Delta x}$$

Implémentation

```
X = linspace(0,L,nx+1)
Uo  = (X/L)**2 * (1-X/L)
Uoo = (2*Uo + (beta**2)*D @ Uo)/2
for t in range(nt):
  U = 2*Uo + (beta**2)*D @ Uo - Uoo
  Uoo = Uo
  Uo = U
```

Conditions intiales

$$u(x,0) = u_0\left(1 - \frac{x}{L}\right)\left(\frac{x}{L}\right)^2$$

$$\frac{\partial u}{\partial t}(x,0) = 0$$

Analyse de stabilité

Amplitude quelconque de la perturbation

Nombre imaginaire

$$U_i^n = U^n e^{ikX_i}$$

k quelconque

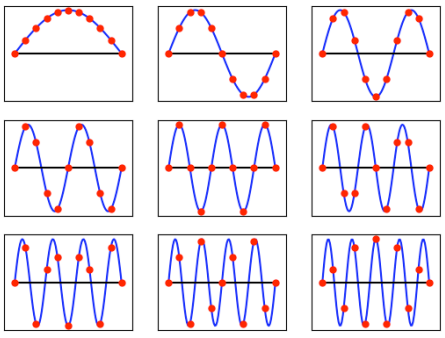
Indice spatial

Considérons une perturbation quelconque de la forme suivante et analysons son évolution….
On souhaite que son amplitude diminue.

Quelques k bien choisis…

m = 11

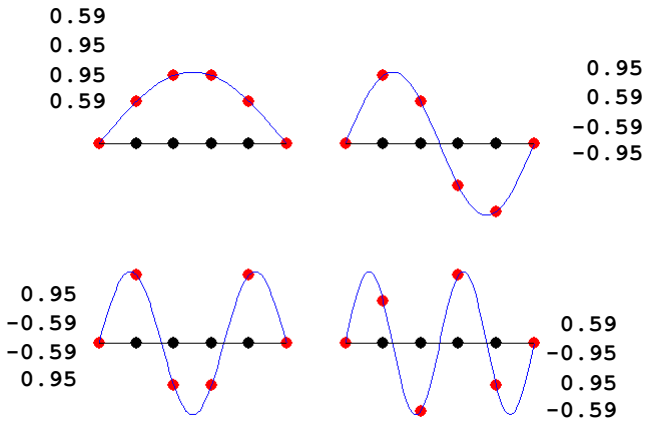
$$U_i^n = U^n \sin(\frac{\hat{k}\pi X_i}{L})$$



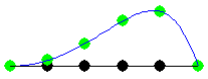
$$\Delta x = 0.1, \Delta t = 0.005$$

Un espace discret de dimension 4

```
X = linspace(0,1,6)
for n=1:4
  U = sin(n *pi * X);
  U(2:5)
end
```



Soit un vecteur tout à fait quelconque de valeurs nodales …

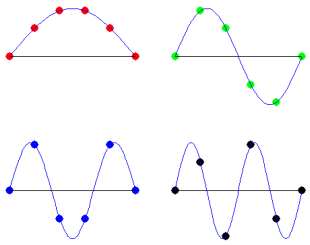


```
x = linspace(0,1,100); u = 2*x.^2.*(1-x.^4);
X = linspace(0,1,6);   U = 2*X.^2.*(1-X.^4);

plot(X,zeros(size(X)),'.k',...
     X,U,'.g',...
     x,u,'-b',...
     [0 1],[0 0],'-k','Markersize',30);
```

N'importe quel vecteur est une combili de sin(nπ)

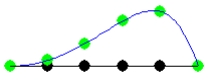
```
A = sin(X(2:5)'*[1:4]* pi)
U = 2* X.^2.*(1- X.^4);
alpha = A \ U(2:5)'
```

```
A' =   0.5878   0.9511   0.9511   0.5878
       0.9511   0.5878  -0.5878  -0.9511
       0.9511  -0.5878  -0.5878   0.9511
       0.5878  -0.9511   0.9511  -0.5878
```

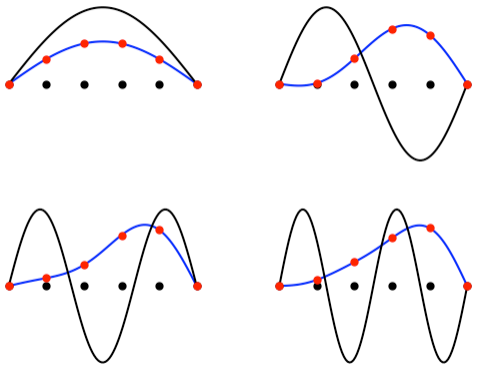


```
alpha =   0.5535
         -0.3311
          0.0972
         -0.0391
```

Et je peux trouver…



```
uh = zeros(100,1)';
for i=1:4
 phi = sin(i*pi*x);
 uh = uh + alpha(i)*phi;
 plot(x,uh,'-b',x,phi,'-k');
end
```



```
alpha =   0.5535
         -0.3311
          0.0972
         -0.0391
```

… cette combili de sin(nπ)

Propagation des erreurs

$$
\begin{aligned}
U_i^{n+1} &= 2U_i^n + \beta^2\left(U_{i+1}^n - 2U_i^n + U_{i-1}^n\right) - U_i^{n-1} \\
&= U_i^n\left(2 + \beta^2\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right)\right) - U_i^{n-1} \\
&= U_i^n\left(2 + \beta^2\left(2\cos(k\Delta x) - 2\right)\right) - U_i^{n-1} \\
&= U_i^n\left(2 - 4\beta^2 \sin^2\left(\tfrac{k\Delta x}{2}\right)\right) - U_i^{n-1}
\end{aligned}
$$

$$U_i^n = U^n e^{ikX_i}$$

Commentaire :
L'analyse de stabilité est ce qu'il y a de plus joli dans le cours FSAB1104 !

Méthodes à pas liés : l'analyse de stabilité nécessite la résolution d'un polynôme…

$$U_i^{n+1} = U_i^n\left(2 - 4\beta^2 \sin^2\left(\tfrac{k\Delta x}{2}\right)\right) - U_i^{n-1}$$

$$U^2\, U_i^{n-1} = U\, U_i^{n-1}\left(2 - 4\beta^2 \sin^2\left(\tfrac{k\Delta x}{2}\right)\right) - U_i^{n-1}$$

$$U^2 = 2U\underbrace{\left(1 - 2\beta^2 \sin^2\left(\frac{k\Delta x}{2}\right)\right)}_{a} - 1$$

Un peu d'algèbre …

$$U^2 - 2aU + 1 = 0$$

$$U = a \pm \sqrt{a^2 - 1}$$

$a^2 = 1$: une racine réelle unique et unitaire
$a^2 < 1$: deux racines complexes conjuguées de module unitaire
$a^2 > 1$: deux racines réelles distinctes (une supérieure à un, une inférieure à un)

Condition de stabilité

$$a^2 \leq 1$$

$$\left(1 - 2\beta^2 \sin^2(\frac{k\Delta x}{2})\right)^2 \leq 1$$

$$1 - 4\beta^2 \sin^2(\frac{k\Delta x}{2}) + 4\beta^4 \sin^4(\frac{k\Delta x}{2}) \leq 1$$

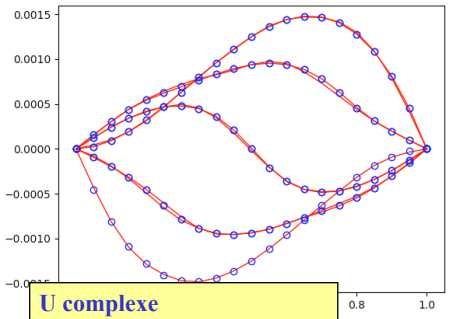
$$-1 + \beta^2 \sin^2(\frac{k\Delta x}{2}) \leq 0$$

… et un peu de calcul

$$\beta = \frac{c\Delta t}{\Delta x} \leq 1$$

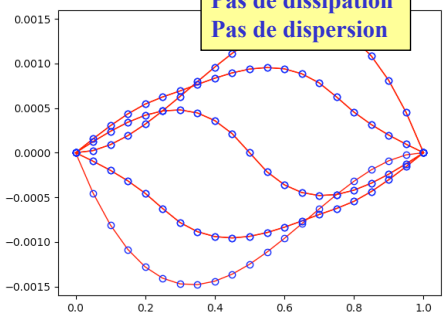
Condition de stabilité pour une équation d'onde….

$\beta = 0.5$



U complexe
Pas de dissipation
Mais présence de dispersion

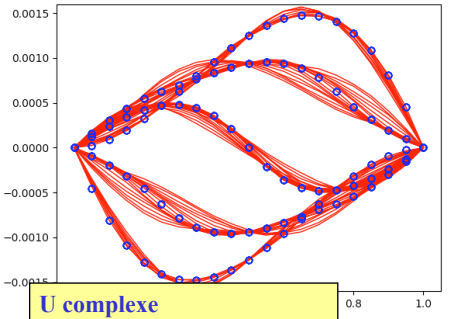
U=1 réel
Pas de dissipation
Pas de dispersion



$\beta = 1.0$

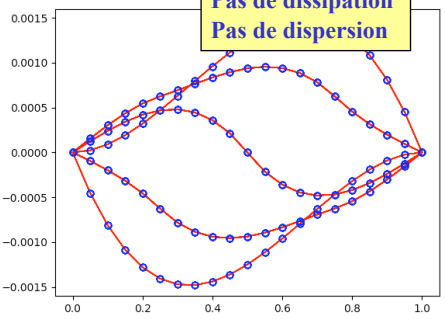
Résultat sur <u>une</u> période

$\beta = 0.5$



U complexe
Pas de dissipation
Mais présence de dispersion

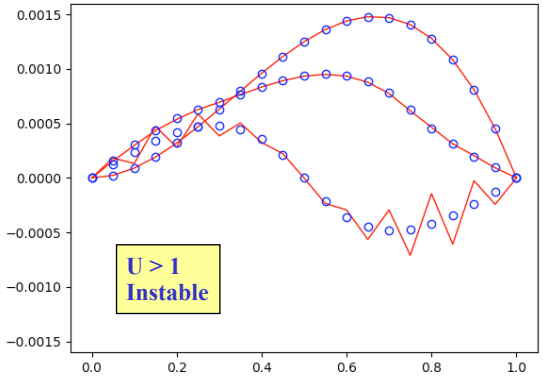
U=1 réel
Pas de dissipation
Pas de dispersion



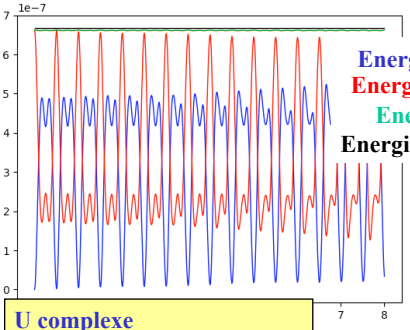
$\beta = 1.0$

Résultat sur <u>huit</u> périodes

$\beta = 1.1$

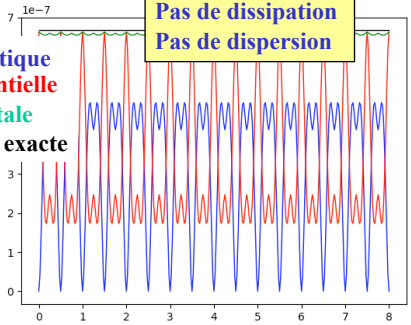


$\beta = 0.5$



Energie cinétique
Energie potentielle
Energie totale
Energie totale exacte

U=1 réel
Pas de dissipation
Pas de dispersion

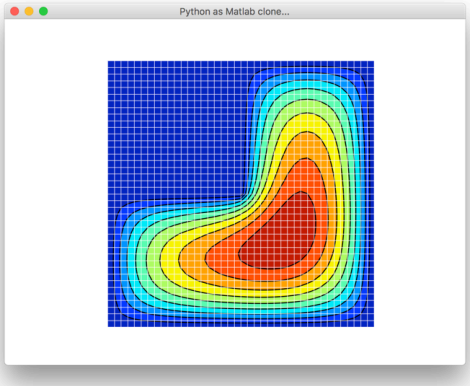


U complexe
Pas de dissipation
Mais présence de dispersion

$\beta = 1.0$

Résultat sur <u>huit</u> périodes

Les méthodes numériques en 18-19 !



C'est presque fini….

C'est presque fini !
Remise du dernier problème : le dimanche 26 mai à 23h59 !
Il est toutefois possible de le faire ce soir !

Et l'examen, Monsieur :-)

Question python
Un programme de 10 lignes à écrire
Les programmes des transparents sont supposés compris
Les 11 problèmes python sont supposés compris

Question « application de l'acquis »
Exercice simple fortement inspiré des exercices simples des notes

Question « compréhension »
Exercice plus subtil dû à la créativité de l'enseignant

Pas de calculatrice,
Formulaire manuscrit recto-verso,
Les travaux python interviendront pour 10% de la note de l'examen.