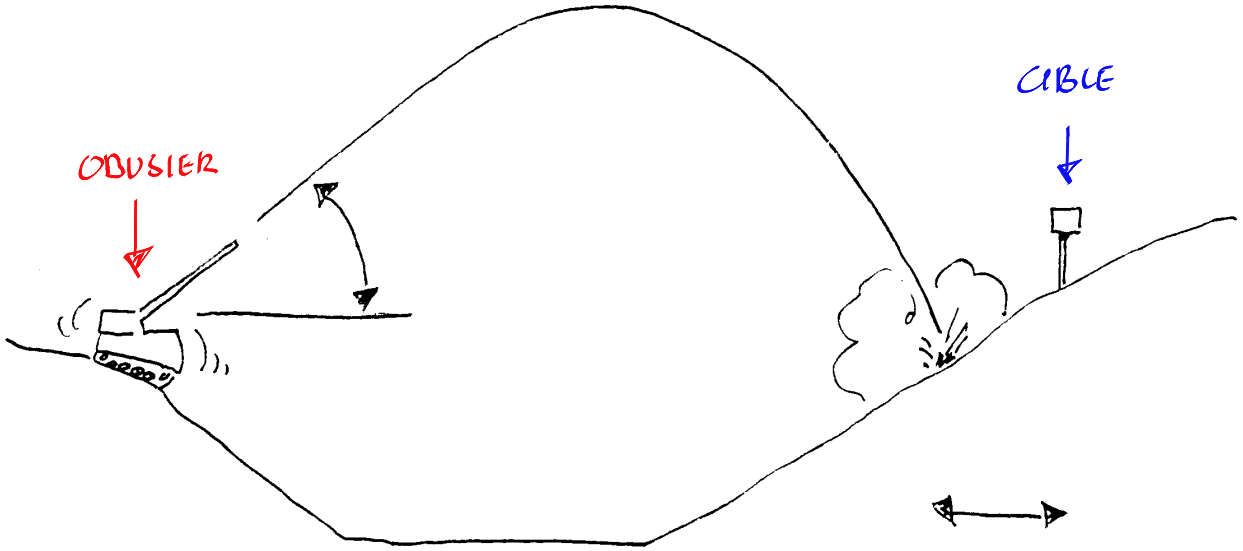
CODE ORANGE : COURS 10

COUR 11 : CODE ARC-EN-CIEL

COURS 12 : FOIRE TOTALE

A quoi servent les méthodes numériques ?



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 ?

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

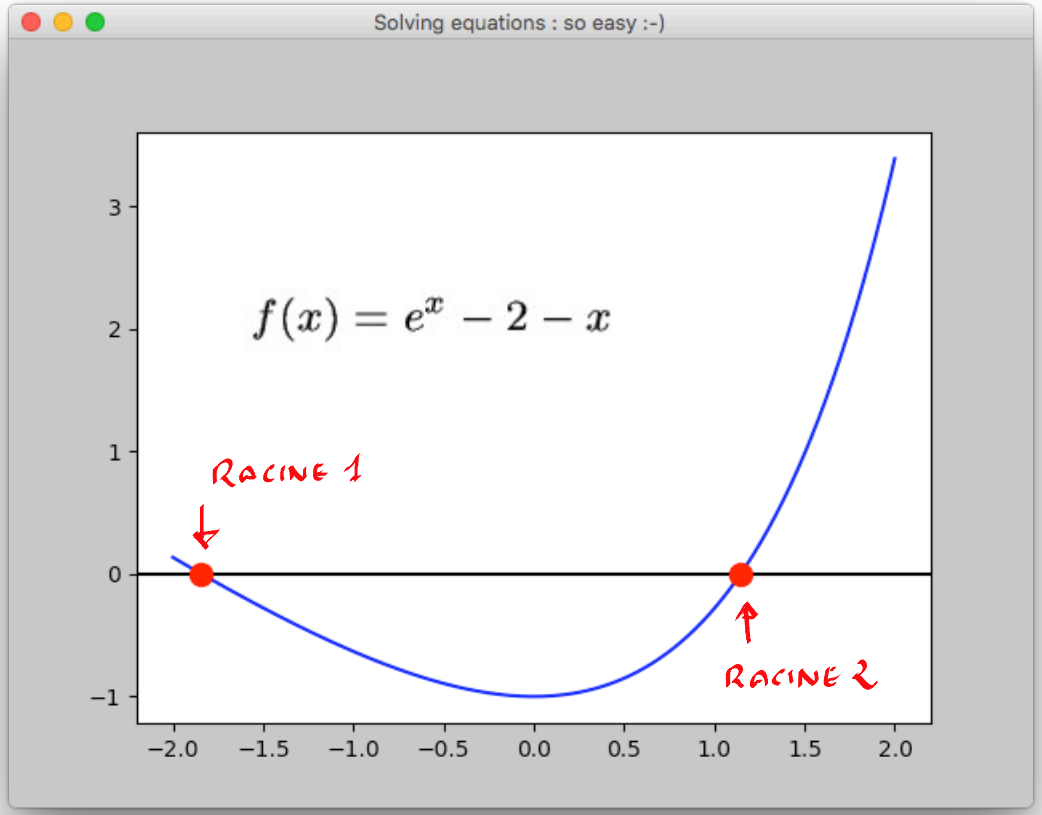
Questions

- Convergence vers une racine
- Candidat initial
- Encadrement

Méthodes numériques itératives

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

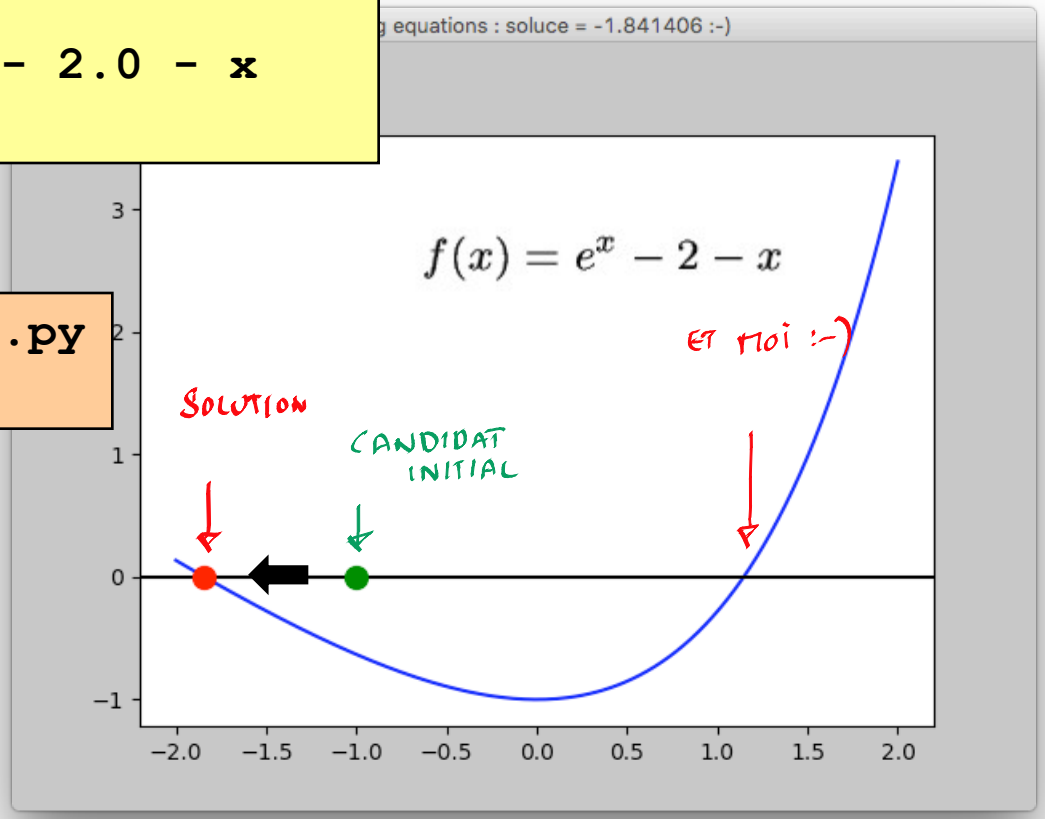
Exemple simple



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]
```

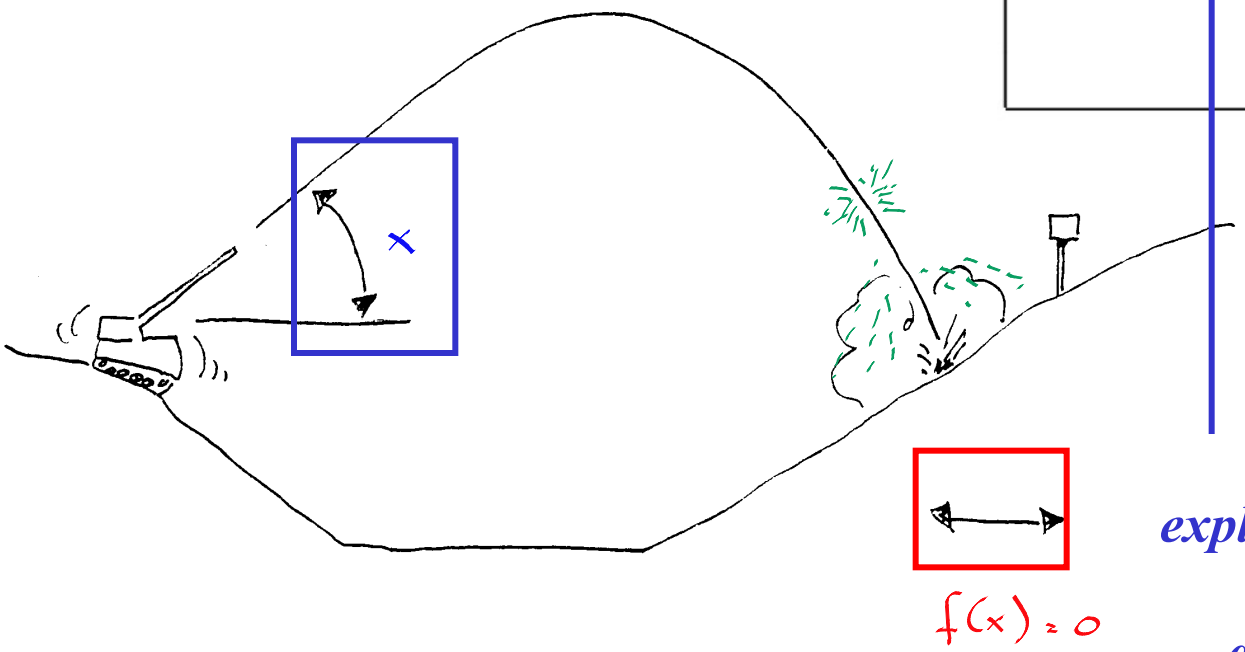


Exemple plus réaliste

Distance entre l'explosion et la cible

Trouver x tel que

$$f(x) = 0$$



explosif fusant charge 3
inclinaison 249
déplacement 4559

Comment prédire ou prévoir f ?

Trouver x tel que

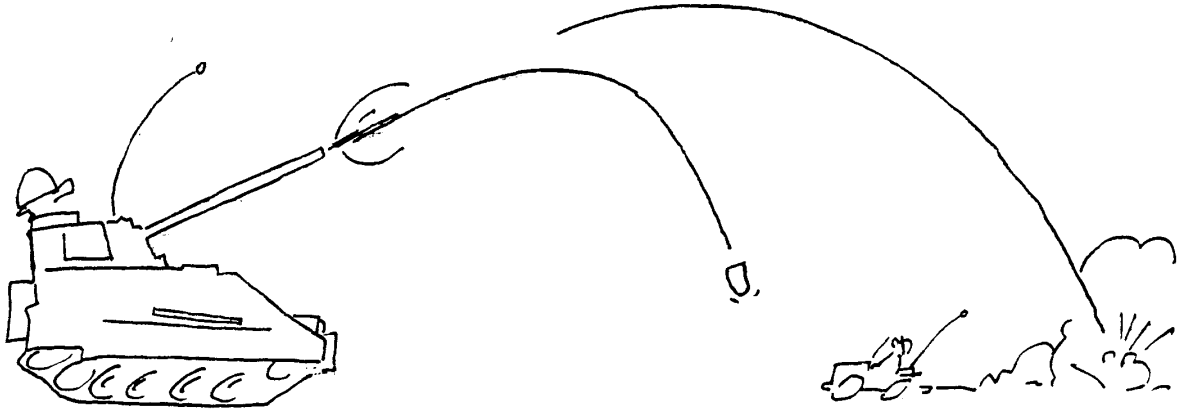
$$\boxed{f(x)} = 0$$

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

$C = 1/2$:-)

C constante positive et inférieure à l'unité

10^{-1}
10^{-2}
10^{-4}
10^{-8}
10^{-16}

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

taux de convergence

ERREUR ITERATION i

CANDIDAT i

$$e_i = x - x_i$$

SOLUTION

10^{-2}
$10^{-2}/2$
$10^{-2}/4$

La séquence converge si on peut trouver C et r

r = taux de convergence

$$\frac{e_i}{(10^{-2})^2} = 1/2$$

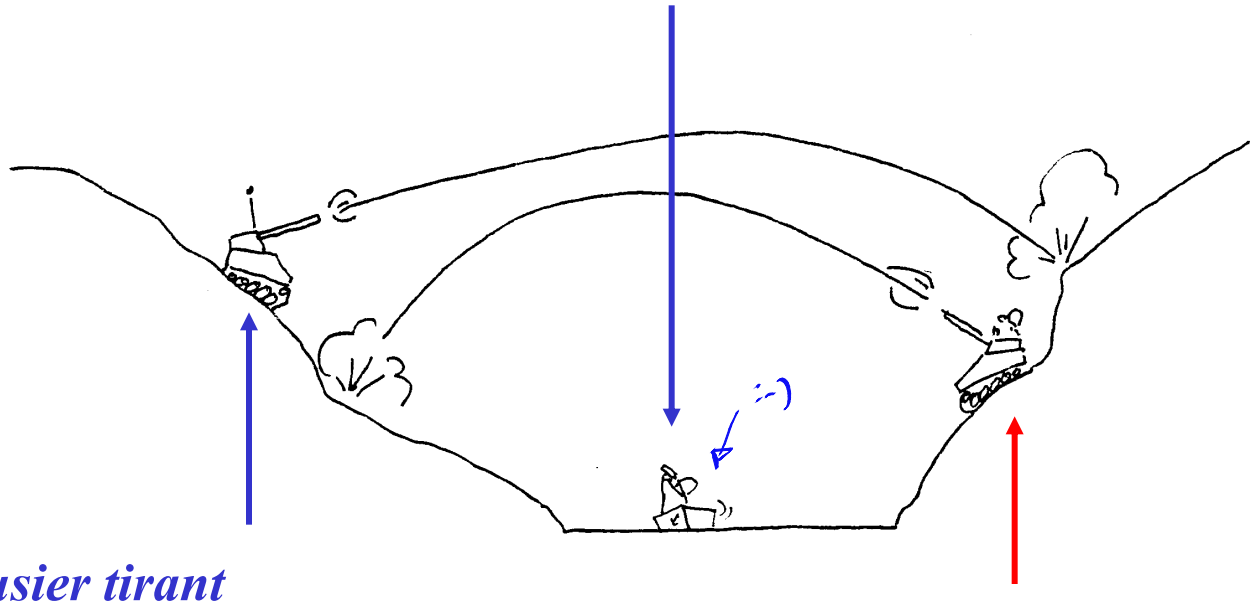
r=1 taux de convergence linéaire

r=2 taux de convergence quadratique

10^{-2}
$10^{-4}/2$

Est-il important de converger rapidement ?

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



Obusier tirant
sur l'obusier adverse…

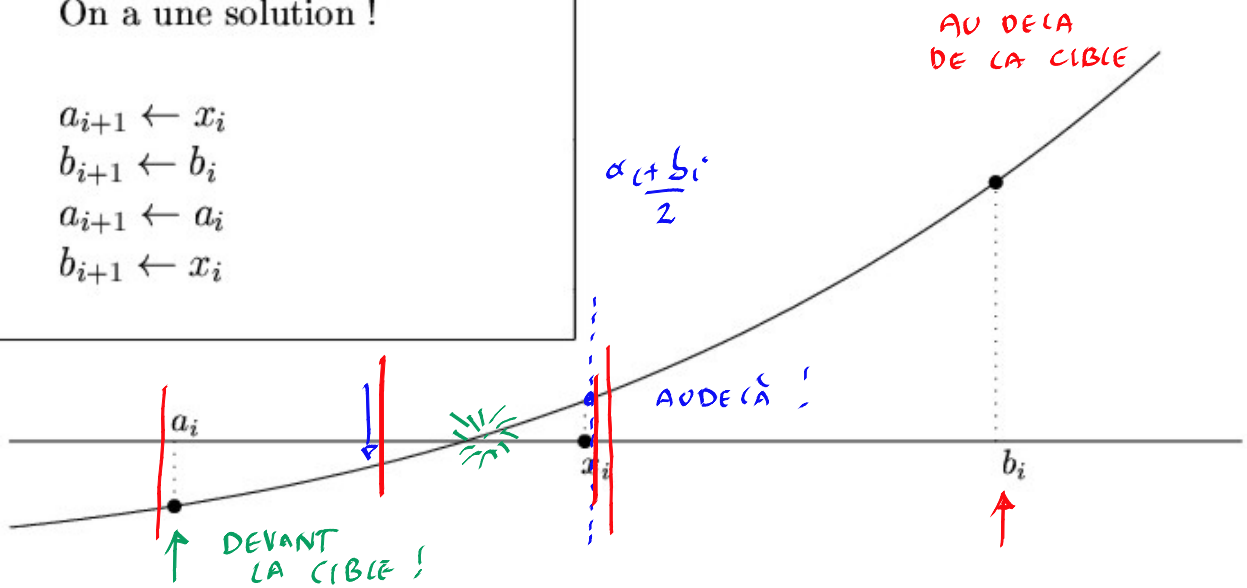
Obusier adverse tentant de
vous détruire au même
moment…

La technique de bissection est une méthode d'encadrement

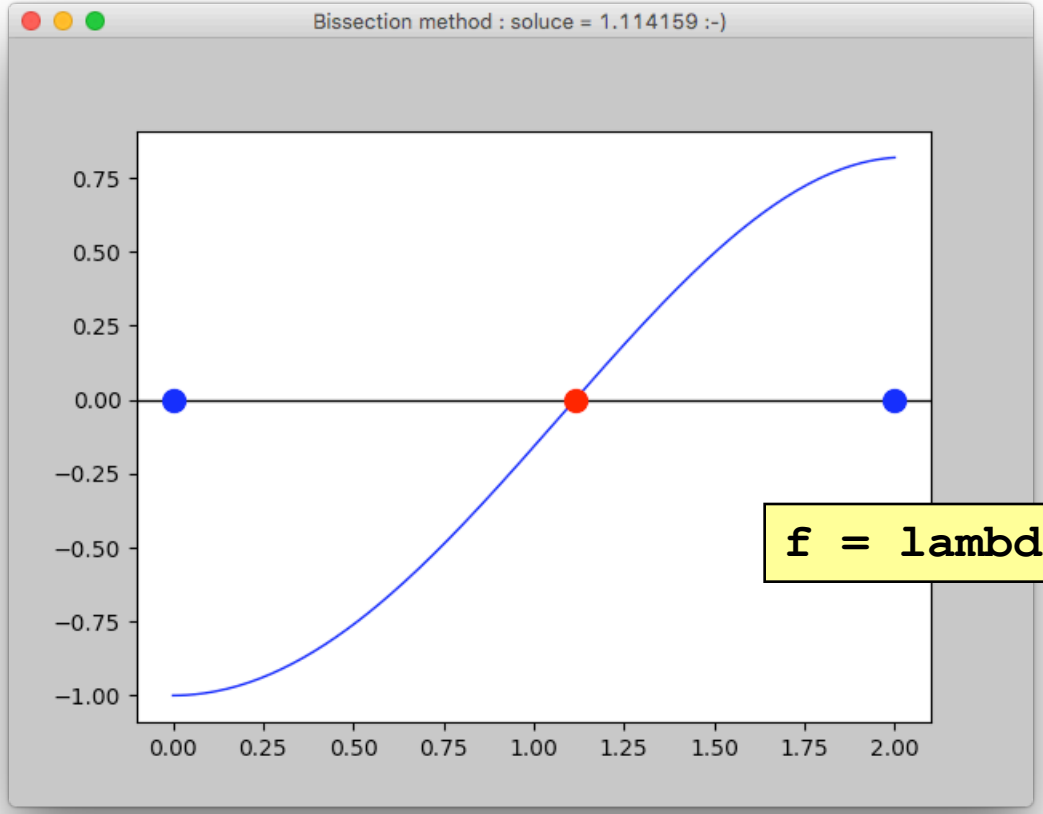
On fournit un intervalle $[a_0, b_0]$ avec $f(a_0)f(b_0) < 0$ et on calcule trois suites a_i, b_i, x_i par

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

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	$a_{i+1} \leftarrow a_i$ $b_{i+1} \leftarrow x_i$

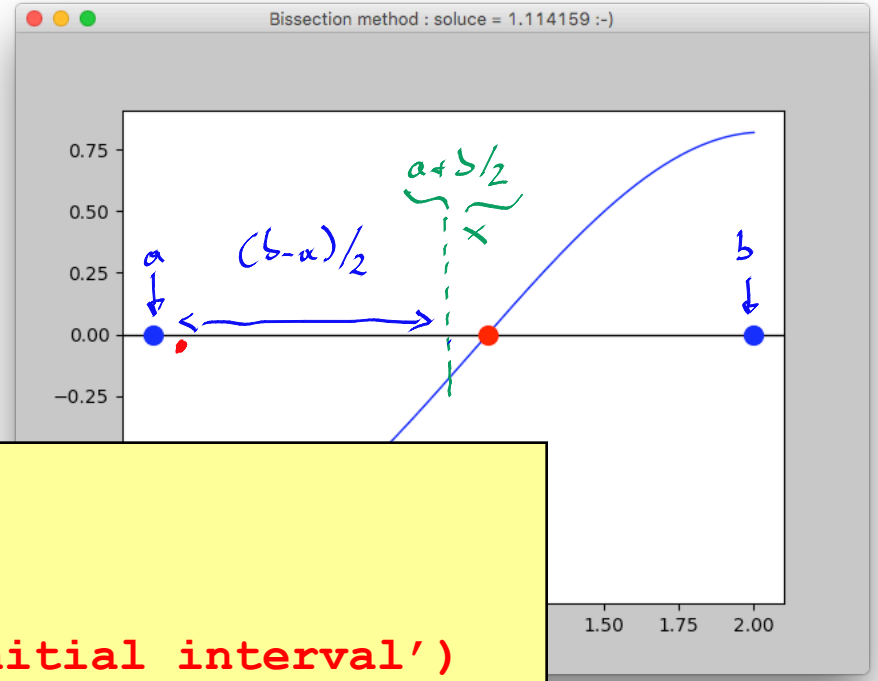


Exemple pour la méthode de bissection…




```
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
```

ROBUSTE !

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

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

$$\Big\downarrow \text{ En effectuant une nouvelle itération}$$

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

$$\Big\downarrow$$

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

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

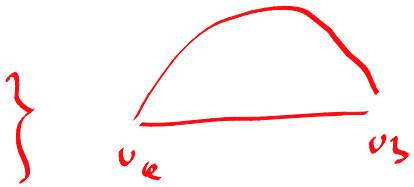
Problème aux conditions aux limites

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

$$r \; : \; \underbrace{(v^a)}_{\text{VITESSE}} \; \rightarrow \; \underbrace{u^b - u^h(b, v^a)}$$

Pour la trajectoire d'un vrai obus…

$$m\frac{dx^2}{dt} = -\gamma \overbrace{\frac{dx}{dt}}^{v_x}$$

$$m\frac{dy^2}{dt} = -mg - \gamma \underbrace{\frac{dy}{dt}}_{v_y}$$

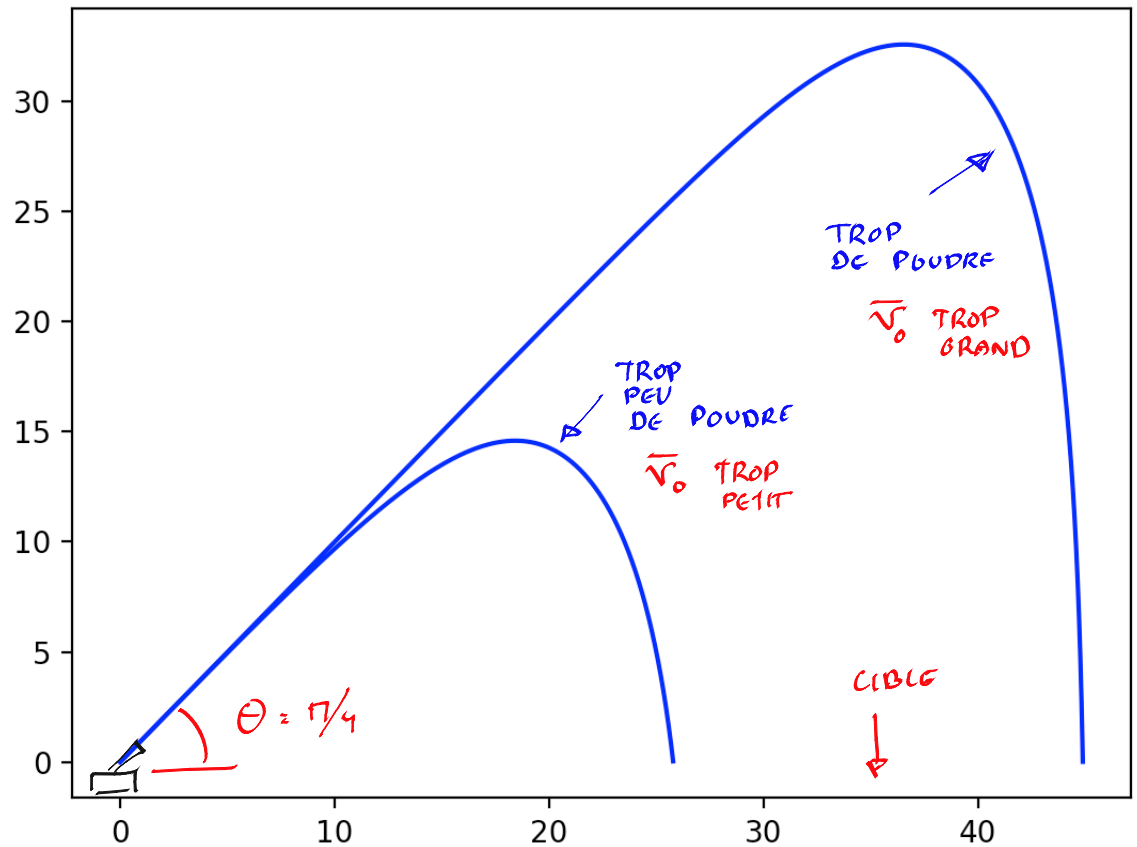
$$\gamma = 0{,}1\sqrt{v_x^2 + v_y^2}$$

$$\frac{dx}{dt} = u$$

$$m\frac{du}{dt} = -\gamma u - mg$$

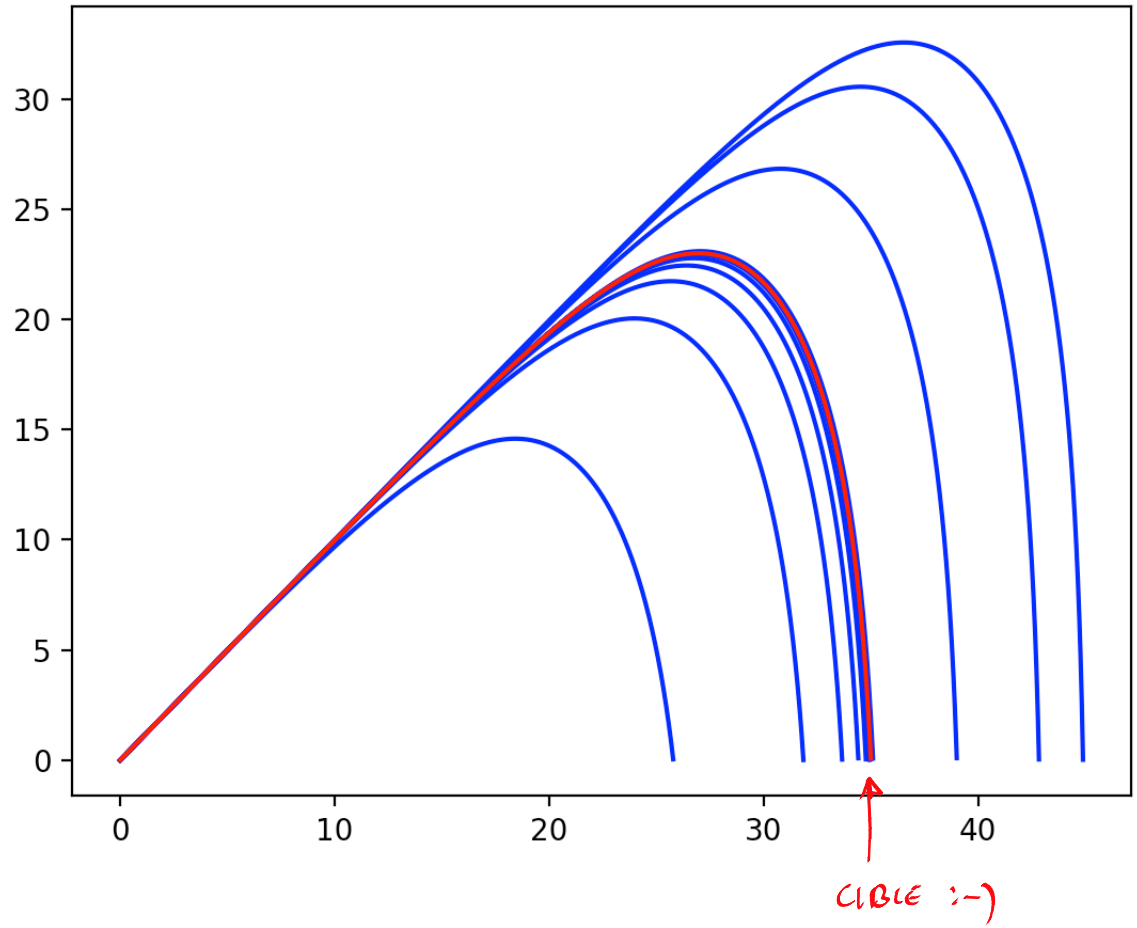
```
def f(u):

  friction = 0.1 * sqrt(u[1]*u[1]+u[3]*u[3])
  mass = 1

  dxdt = u[1]
  dudt = (- friction *u[1])/mass
  dydt = u[3]
  dvdt = -9.81 -(friction * u[3])/mass

  return array([dxdt,dudt,dydt,dvdt])
```

$$\begin{cases} u_0 = \bar{v}_0 \cos(\theta) \\ v_0 = \bar{v}_0 \sin(\theta) \end{cases}$$



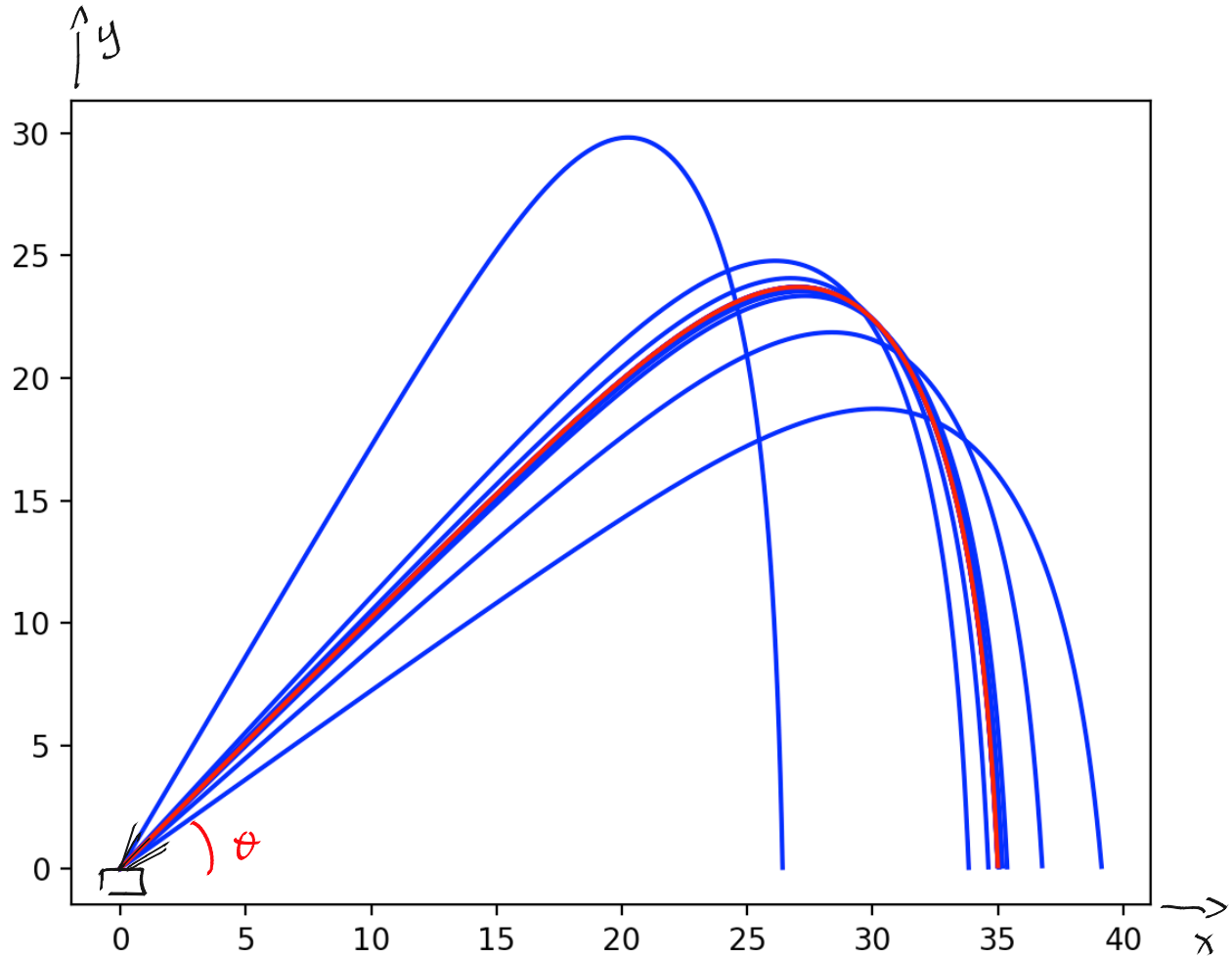
Ajustons la puissance du tir…

Utilisons la technique bissection pour ajuster la vitesse initiale de l'obus.



Ajustons la puissance du tir…

Utilisons la technique bissection pour ajuster la vitesse initiale de l'obus.



Ajustons l'angle du tir…

Utilisons la technique bissection pour ajuster l'inclinaison du tube.

Seconde idée : Différences finies

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

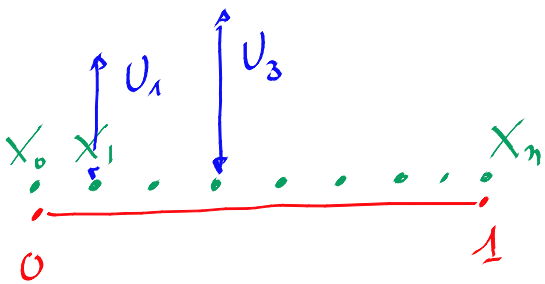
$$u(X_i) \approx u^h(X_i) = U_i$$

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…

$U_0 = u_a$

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

x_1

x_2

$U_m = u^b$

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 !

SOLUTION PARTICULIERE

$$u(x) = x + \underbrace{A\cos(x) + B\sin(x)}_{\text{SOLUTIONS EQUATION HOMOGENE}}$$

Solution analytique

SPARSE

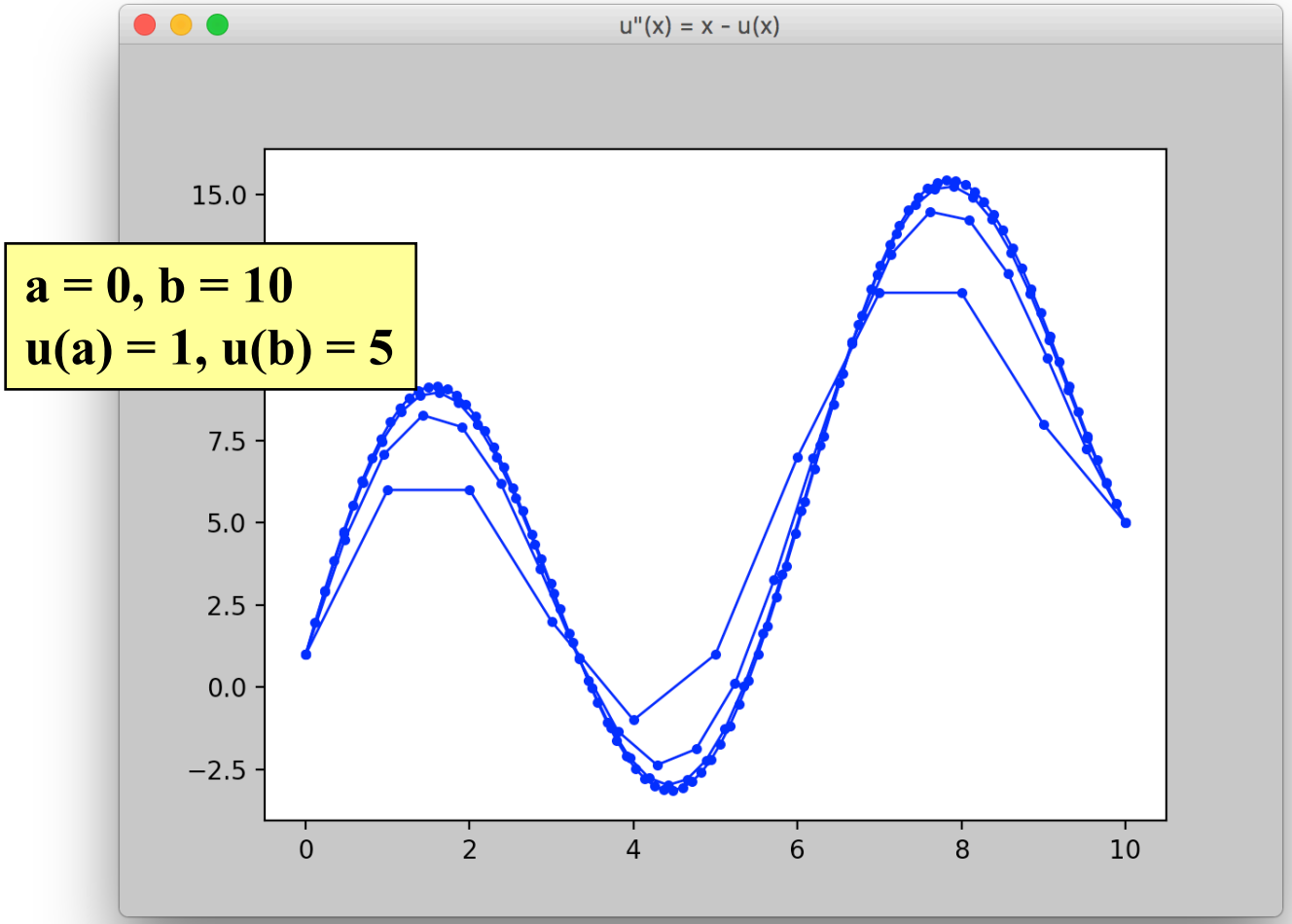
SPSOLVE :-)

Problème numérique

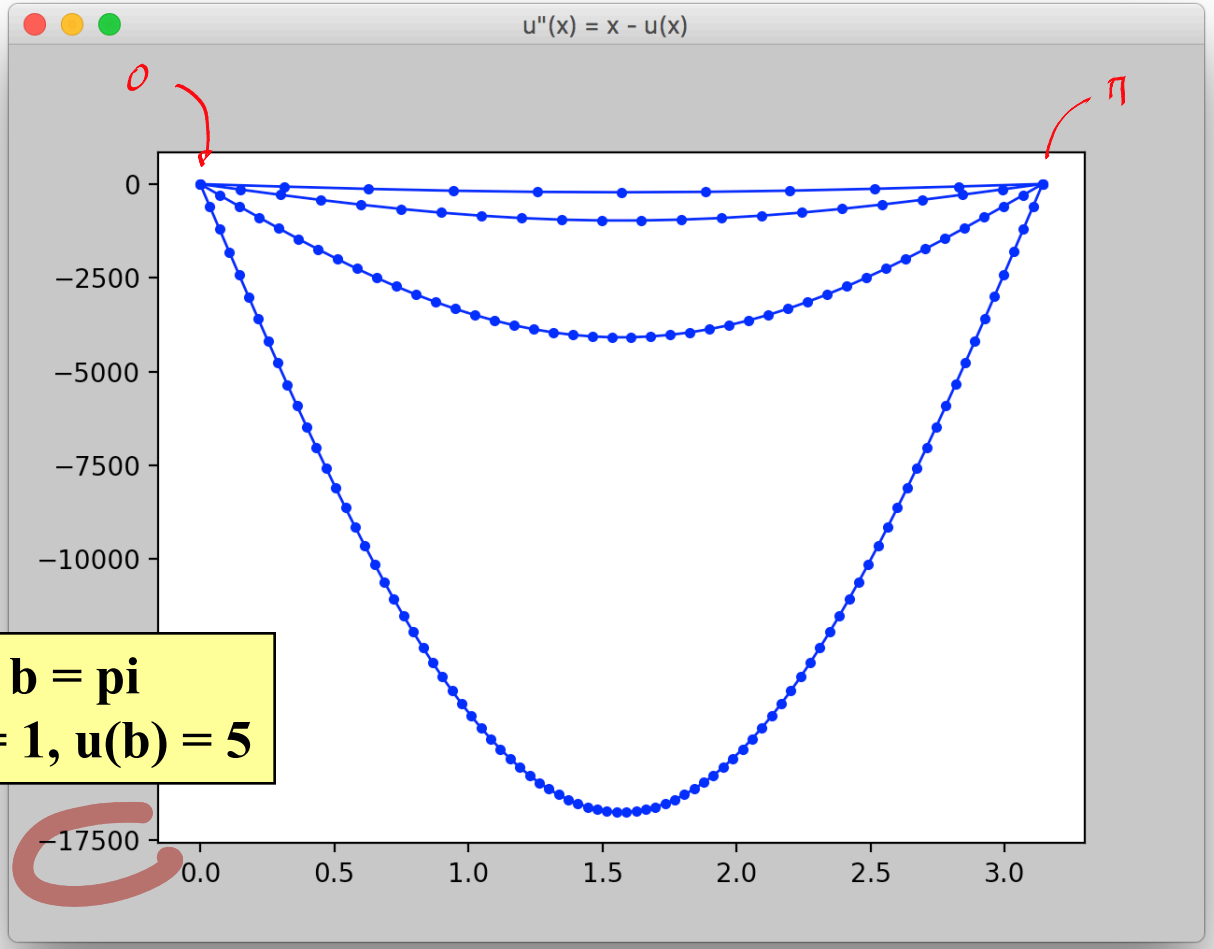
$$\frac{1}{h^2}\begin{bmatrix} 0 & 0 & & & & & & & \\ 1 & -2 & 1 & & & & & & \\ & 1 & -2 & 1 & & & & & \\ & & 1 & -2 & 1 & & & & \\ & & & 1 & -2 & 1 & & & \\ & & & & \cdot & \cdot & \cdot & & \\ & & & & & 1 & -2 & 1 & \\ & & & & & & 1 & -2 & 1 \\ & & & & & & & 0 & 0 \end{bmatrix}\begin{bmatrix} U_0 \\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

```
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 :-)



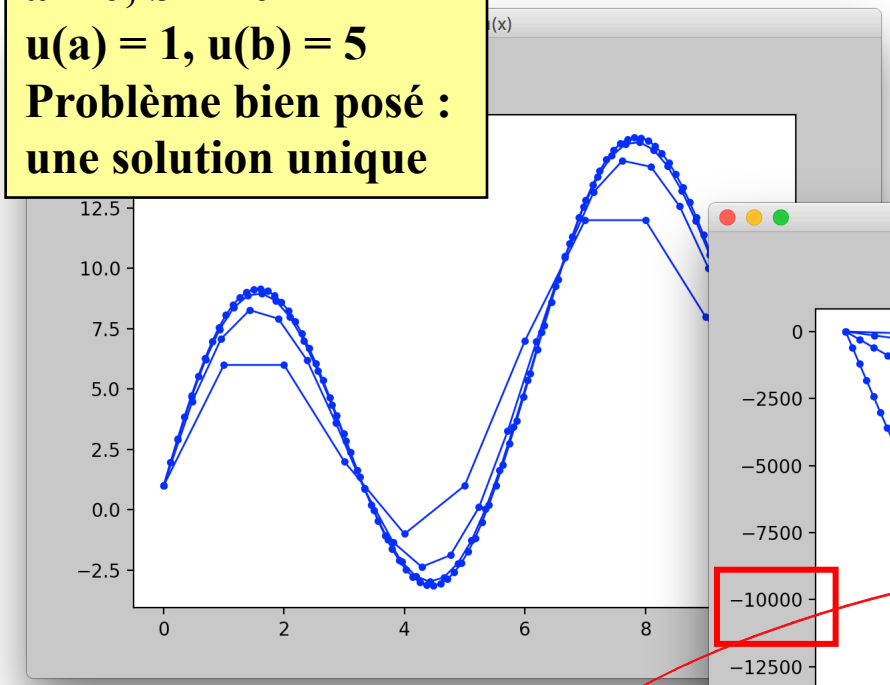
Un autre joli résultat :-(

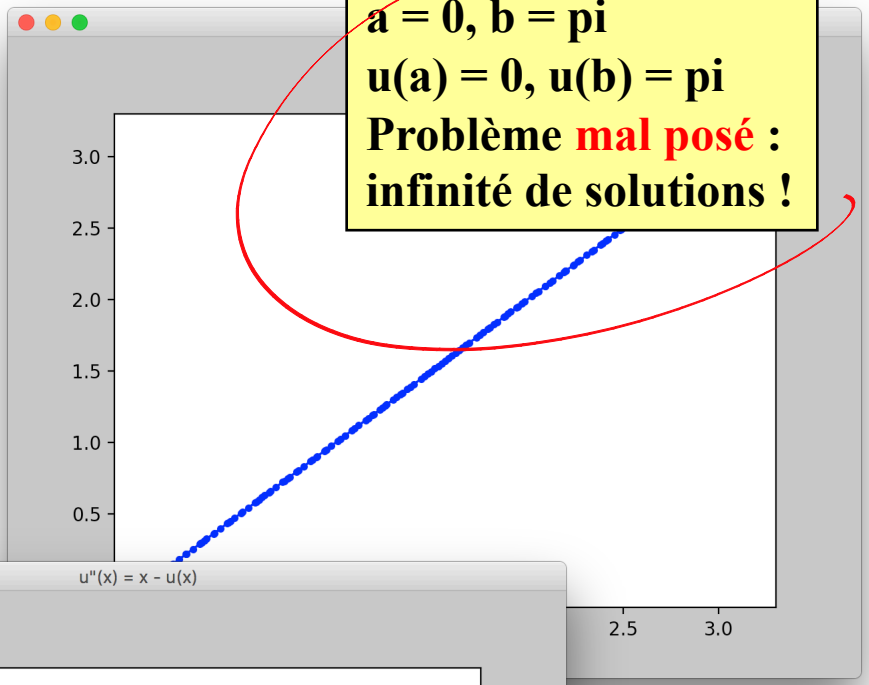


a = 0, b = pi
u(a) = 1, u(b) = 5

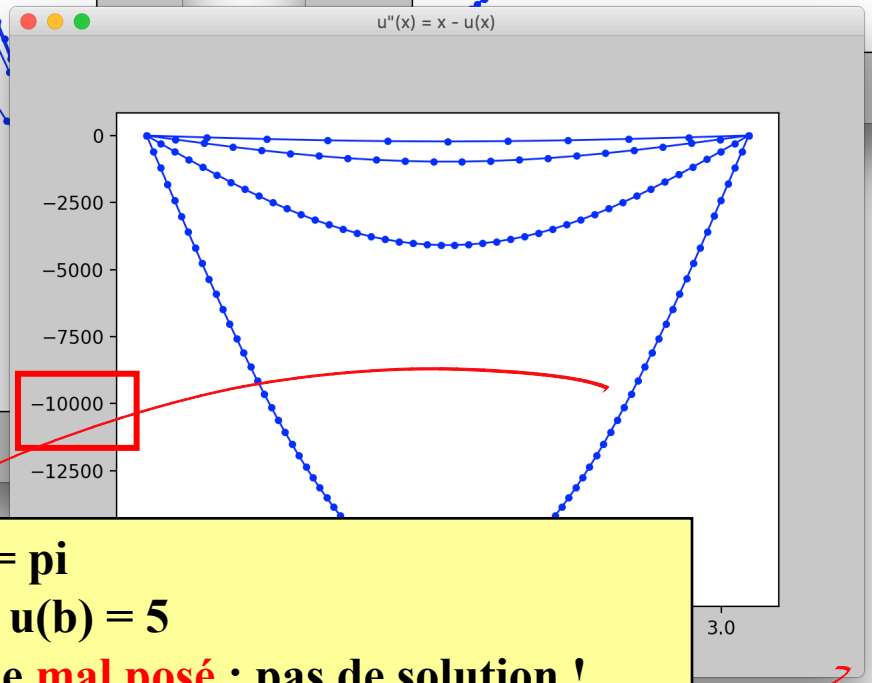
Est-ce que cela marche toujours ?

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

$x^{i+1} = g(x^i)$

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 ?

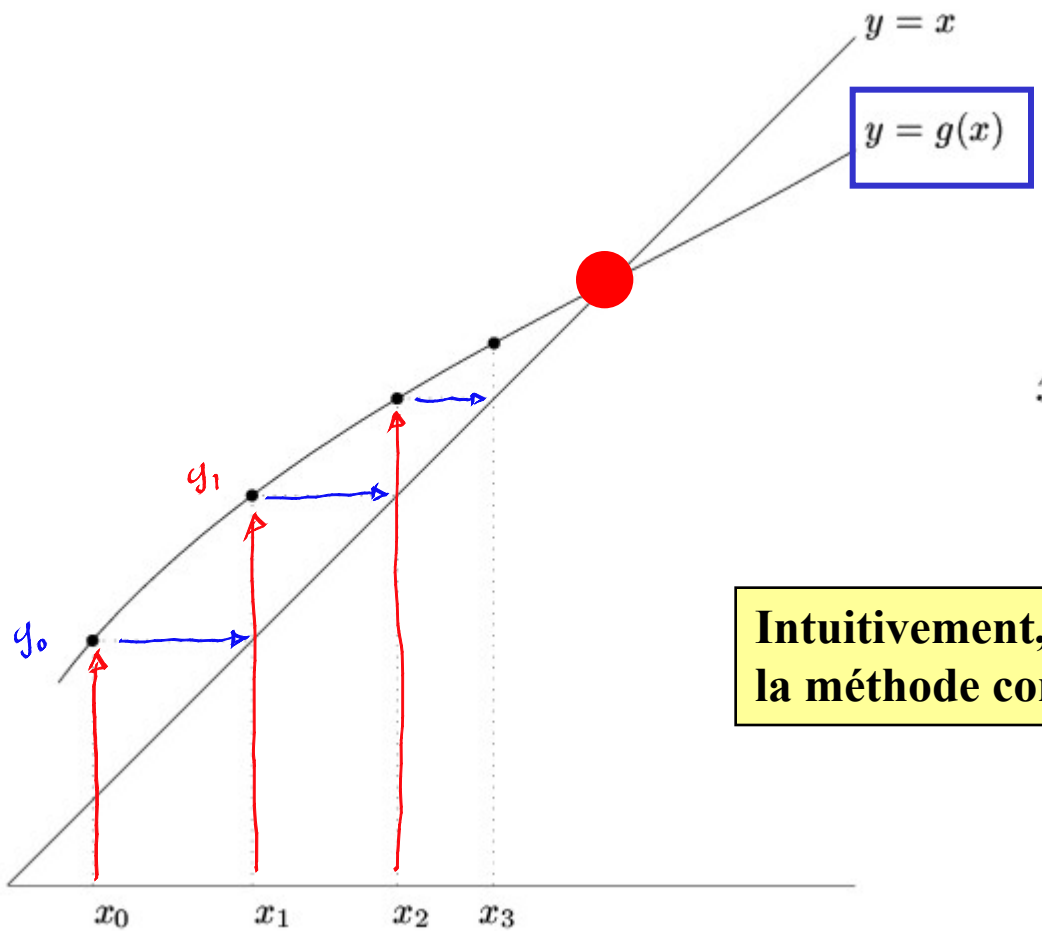
Réécrivons l'équation $f(x) = 0$ sous une forme[2]

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

On peut ainsi écrire $x = \dfrac{(x^3+1)}{3}$, 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

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

$$x = \underbrace{\frac{x^3+1}{3}}_{g(x)}$$

$$x = \overbrace{x^3 - 3x + 1}^{=0} + x$$

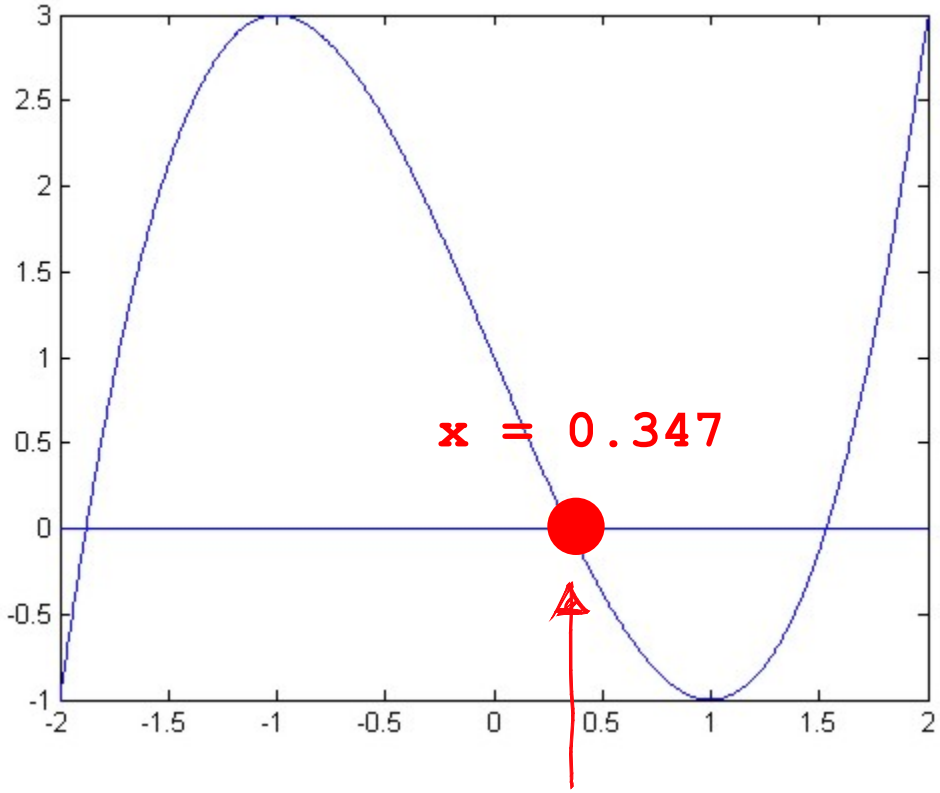
$$x = \underbrace{x^3 - 2x + 1}_{g(x)}$$

$$x = \overbrace{-x^3 + 3x - 1}^{=0} + x$$

$$x = \underbrace{-x^3 + 4x - 1}_{g(x)}$$

$$x = \underbrace{\sqrt[3]{3x-1}}_{g(x)}$$

Exemple

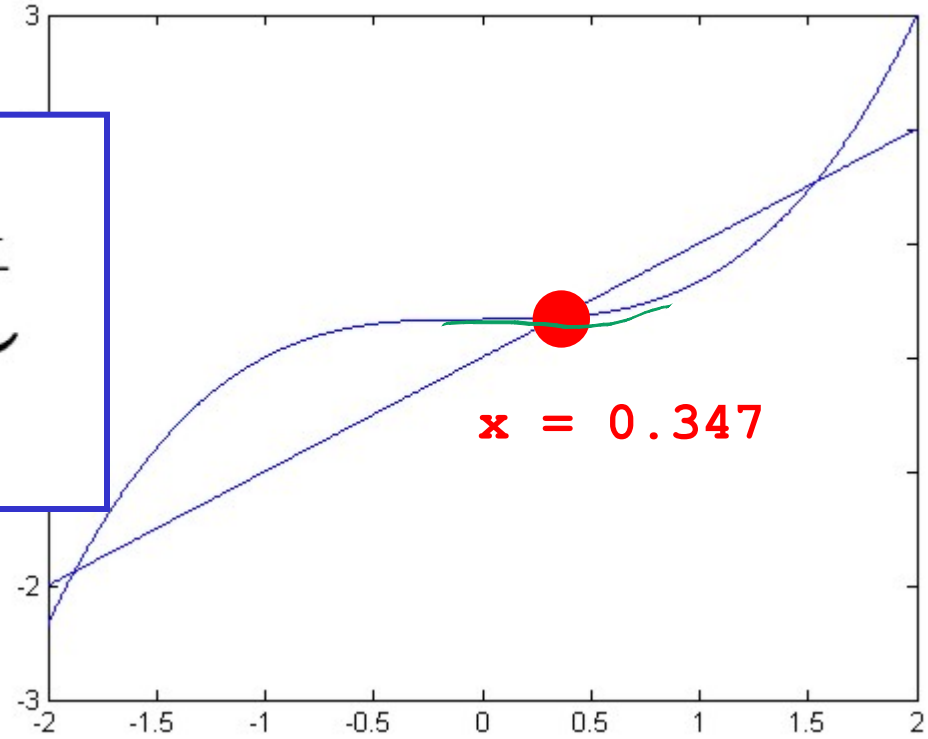


$$\underbrace{x^3 - 3x + 1}_{f(x)} = 0$$

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…

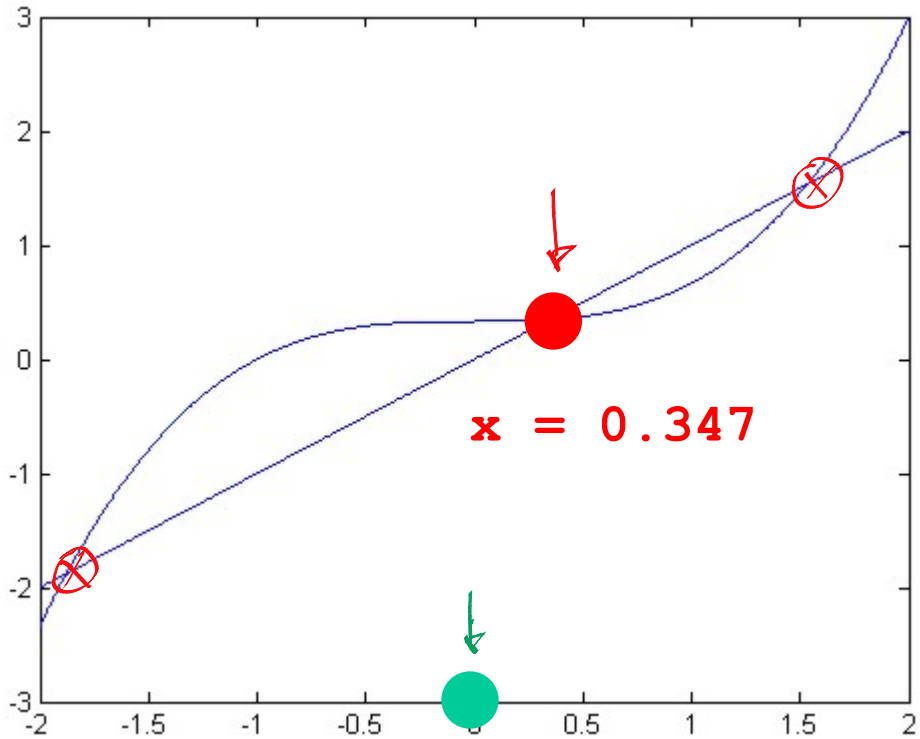
```
def iter(x,tol,nmax,g):
  n = 0; delta = float("inf")
  while abs(delta) > tol and n < nmax :
    n = n + 1
    xold = x
    x = g(x)
    delta = xold - x
    print(" x = %21.14e (%d)" % (x,n))
  return x
```

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

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

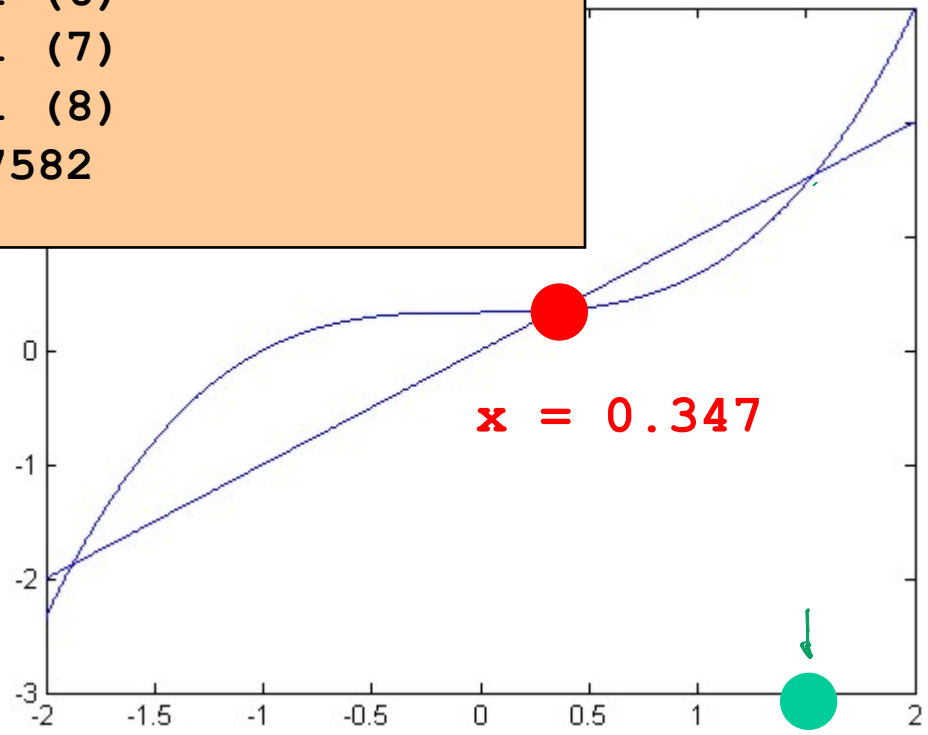
```
>>> print("Found x = ", iter(0.0,10e-3,50,g))
 x =  3.33333333333333e-01 (1)
 x =  3.45679012345679e-01 (2)
 x =  3.47102186947062e-01 (3)
Found x =  0.347102186947062
```

Essayons…



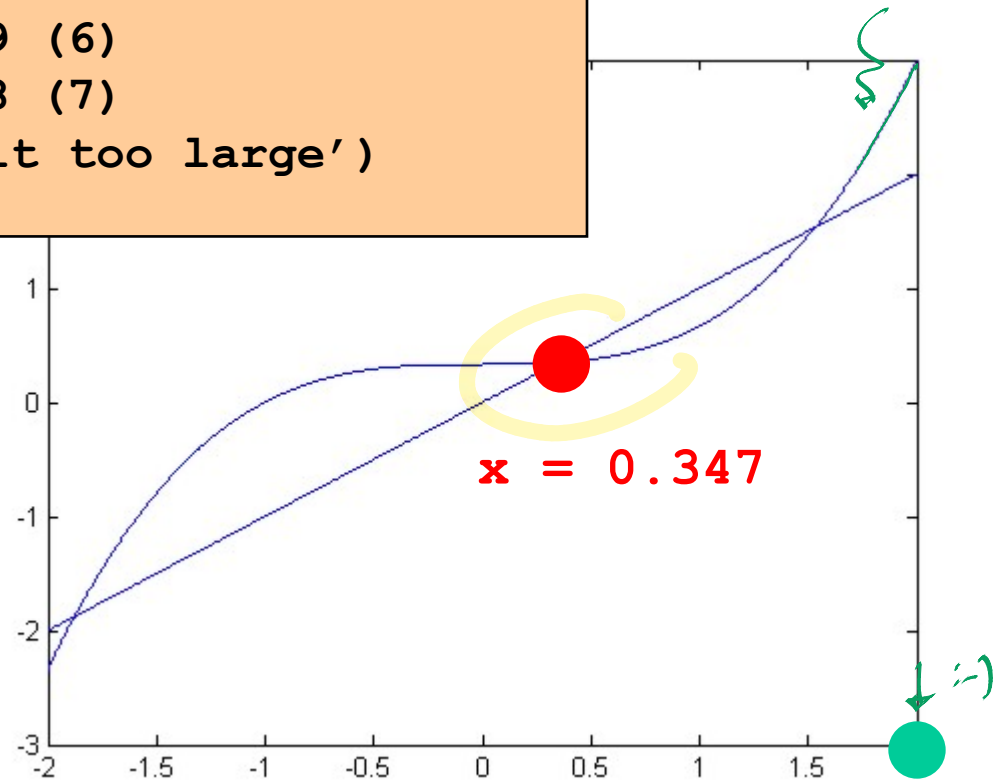
```
>>> print("Found x = ", iter(1.5,10e-3,50,g))
 x =  1.45833333333333e+00 (1)
 x =  1.36716338734568e+00 (2)
 x =  1.18513797762971e+00 (3)
 x =  8.88195982531111e-01 (4)
 x =  5.66896932292182e-01 (5)
 x =  3.94061625221864e-01 (6)
 x =  3.53730562615967e-01 (7)
 x =  3.48086882210758e-01 (8)
Found x =  0.348086882210758
```

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…

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

$$\underbrace{x_{i+1} - x}_{e_{i+1}} = \underbrace{g(x_i) - g(x)}_{g'(\xi)\,(\underbrace{x_i - x}_{e_i})}$$

$$|g'(\xi)| < K$$

$$|e_{i+1}| \leq K\,|e_i|$$

$$\lim_{i\to\infty} |e_i| \leq \underbrace{\lim_{i\to\infty} K^i}_{\to 0 \text{ si } K<1 \text{ :-)}} |e_0|$$

Pour les fonctions lipschitziennes…

$$\underbrace{(x_{i+1} - x)}_{e_{i+1}} = g(x_i) - g(x)$$

En vertu du théorème de la moyenne.

$$= g'(\xi)\ (x_i - x)$$

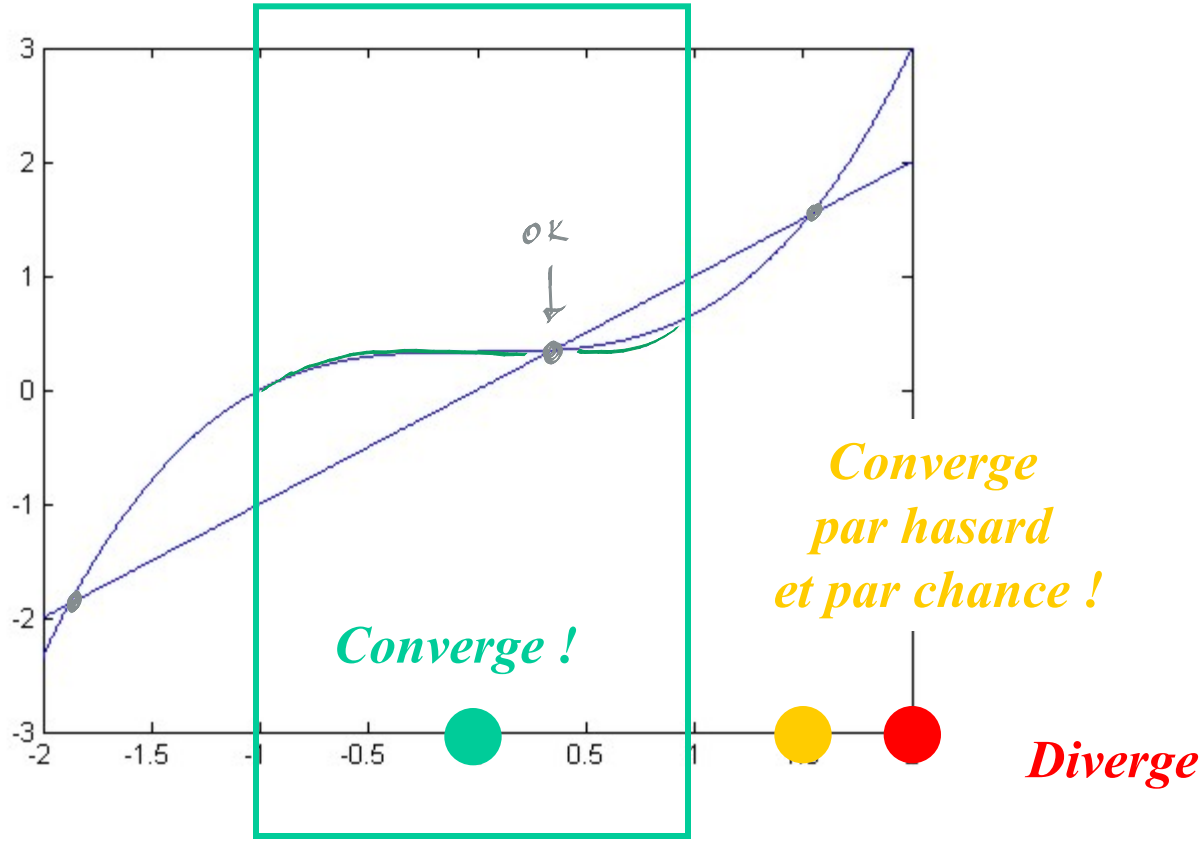
En vertu de la condition de Lipschitz.

$$|e_{i+1}| \leq K\ |e_i|$$

$$\boxed{0 \leq \lim_{i\to\infty} |x_i - x| \leq \lim_{i\to\infty} K^i\ |x_0 - x| = 0}$$

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 !

$$\underbrace{f(x)}_{=0} = f(x_i) + \overbrace{(\underbrace{x}_{x_{i+1}} - x_i)}^{\Delta x} f'(x_i) + \underbrace{\frac{(x-x_i)^2}{2} f''(\xi)}_{\approx 0}$$

$$f'(x_i)\,\Delta x = -f(x_i)$$

$$\boxed{\begin{aligned} \Delta x &= -\frac{f(x_i)}{f'(x_i)} \\ x_{i+1} &= x_i + \Delta x \end{aligned}}$$

METHODE
DE NEWTON
RAPHSON :-)

Méthode de Newton-Raphson

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2}f''(x_0) + \dots$$

On fournit x_0

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

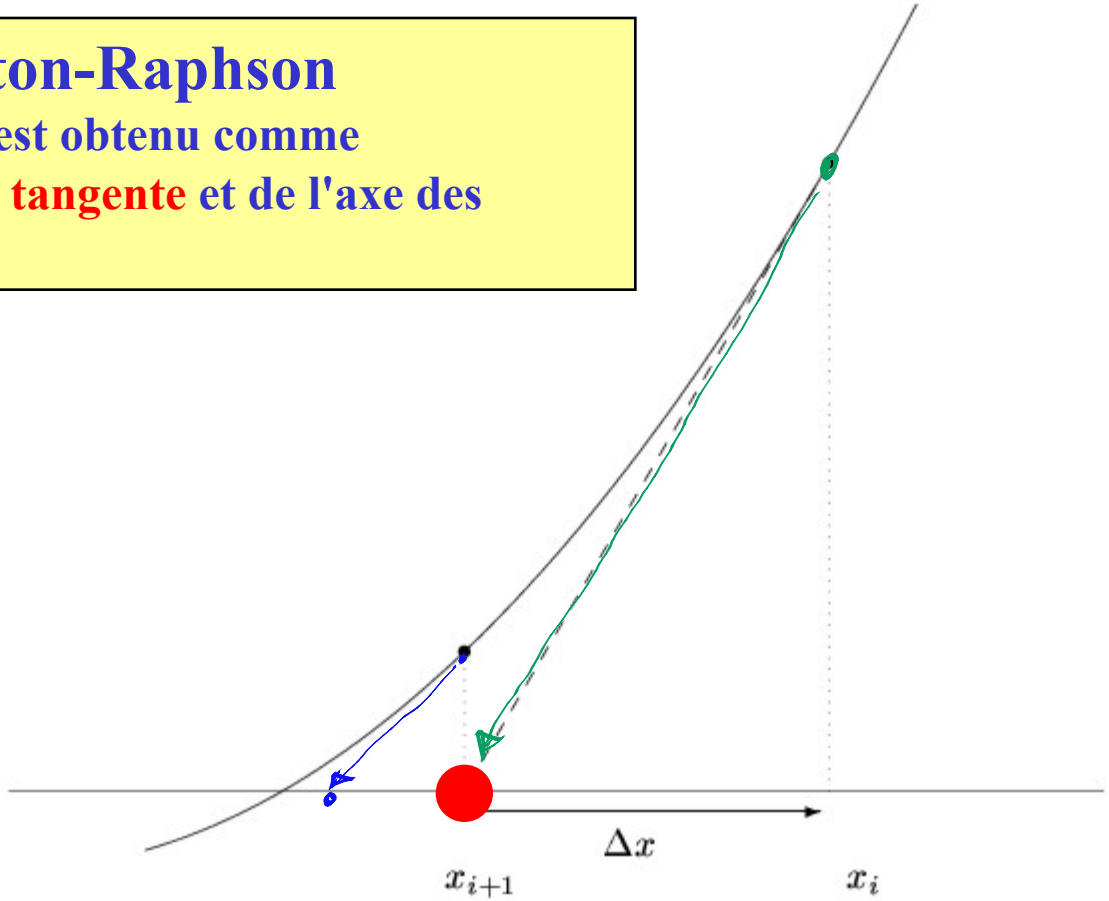
$$\begin{aligned} f'(x_i)\overbrace{(x_{i+1} - x_i)}^{\Delta x} &= -f(x_i) \\ x_{i+1} &= x_i + \Delta x \end{aligned}$$

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



Taux de convergence de Newton-Raphson

$$\underbrace{f(x)}_{=0} = f(x_i) + \underbrace{(x - x_i)}_{e_i} f'(x_i) + \underbrace{\frac{(x-x_i)^2}{2}}_{\frac{e_i^2}{2}} f''(\xi)$$

$$0 = f + e_i f' + e_i^2 \frac{f''}{2}$$

$$\boxed{f = -e_i f' - e_i^2 \frac{f''}{2}}$$

$$e_{i+1} = x - x_{i+1}$$

$$= \underbrace{x - (x_i}_{e_i} - \frac{f}{f'})$$

$$\boxed{e_{i+1} = \frac{e_i f' + f}{f'}}$$

$$e_{i+1} = \frac{\cancel{e_i f'} - \cancel{e_i f'} - e_i^2 f''/2}{f'}$$

$$e_{i+1} = e_i^2 \underbrace{\left[-\frac{f''}{2f'} \right]}_{C}$$

Newton-Raphson : Taux de convergence quadratique

Propagation de l'erreur dans un schéma de Newton-Raphson

$$
\begin{aligned}
e_{i+1} &= x - x_{i+1} \\
&= x - \left(x_i - \frac{f(x_i)}{f'(x_i)} \right) \\
&= \frac{e_i f'(x_i) + f(x_i)}{f'(x_i)}
\end{aligned}
$$

$$
e_{i+1} = \underbrace{-\frac{1}{2}\frac{f''(\xi)}{f'(x_i)}}_{C} e_i^2
$$

Développement en série de Taylor

$$
\begin{aligned}
0 &= f(x) \\
&= f(x_i) + \underbrace{(x - x_i)}_{e_i} f'(x_i) + \underbrace{(x - x_i)^2}_{e_i^2} \frac{f''(\xi)}{2}
\end{aligned}
$$

Convergence si cette constante est comprise entre [-1,1]…

$$f'(x_i) \approx \frac{f(x_i+h) - f(x_i-h)}{2h}$$

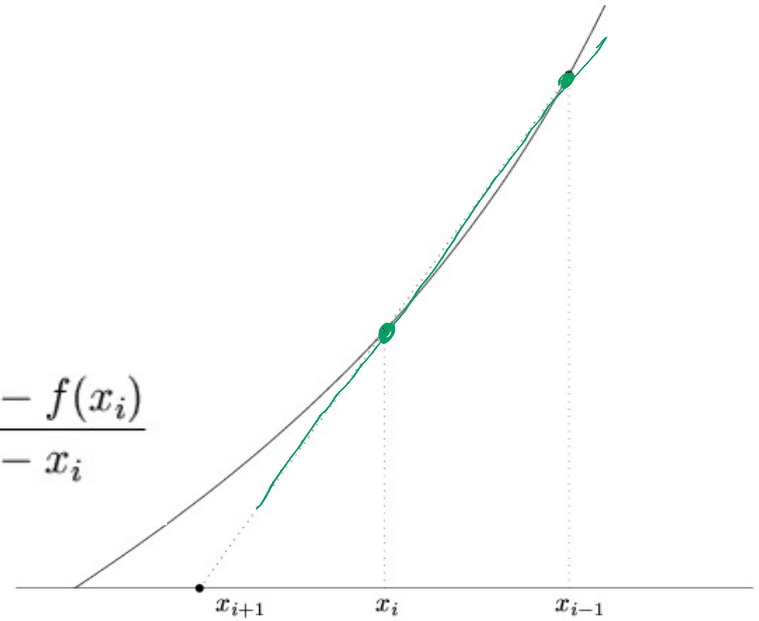
Evaluation numérique de f'

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

Taux de convergence superlinéaire mais pas quadratique !

$$|e_{i+1}| = C|e_i|^{1.618}$$

Méthode de la sécante

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

$$\left\{ \begin{array}{rcl} f_1(x_1, x_2, \ldots, x_n) & = & 0 \\ f_2(x_1, x_2, \ldots, x_n) & = & 0 \\ \ldots & & \\ f_n(x_1, x_2, \ldots, x_n) & = & 0 \end{array} \right.$$

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…