

# Anal avec python

Yehor Korotenko

February 1, 2025

# Contents

<b>1</b>	<b>cours 1</b>	<b>2</b>
1.1	Modèles discrètes . . . . .	2
1.1.1	Modèle de croissance géométrique . . . . .	2
1.2	Modèles continues . . . . .	3
1.2.1	Modèle de Malthus . . . . .	3
1.2.2	Modèle Verhulst . . . . .	4
1.3	Modèle de croissance logistique . . . . .	5
<b>2</b>	<b>cours 2</b>	<b>6</b>
2.1	Notion de champ de vecteurs associée à une EDO . . . . .	6
2.1.1	Généralités et définitions . . . . .	6
2.1.2	Dessins de champs de vecteurs . . . . .	8
2.1.3	Recherche de solution approchée de modèles sous python . . . . .	9
2.2	Modèle de prédateur proie (lotka-voltena (1931)) . . . . .	10
<b>3</b>	<b>cours3:</b>	
	<b>Interpolation polynomiale</b>	<b>11</b>
3.0.1	Valeur ajoutée par l'itération . . . . .	12
3.0.2	Obtenir numériquement la vitesse de convergence . . . . .	13

# Chapter 1

## cours 1

### 1.1 Modèles discretes

On désigne par  $N(t)$  la population d'individus à l'instant  $t$ .

Équation du modèle discret:

$$\underbrace{N(t + \Delta t) - N(t)}_{\text{variation de la population}} = \underbrace{n}_{\text{nombre de naissances}} - \underbrace{m}_{\text{nombre de décès}} + \underbrace{i}_{\text{immigration}} - \underbrace{e}_{\text{émigration}}$$

sol de migration

#### 1.1.1 Modèle de croissance géométrique

- hypothèse:

- solde migration nul: i.e  $i - e = 0$

- nombre de croissance proportionnel à la taille de la population  $n = \lambda \Delta t N(t)$   
taux de natalité

- Idem pour le nombre de décès:  $m = \mu \Delta t N(t)$  taux de mortalité

- Modèle: On pose  $N_n = N(t_n)$  la taille de la population à l'instant  $t_n$ .

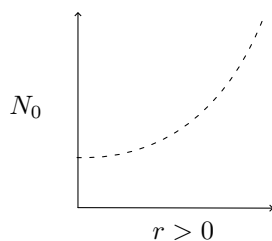
$$N_{n+1} - N_n = \lambda \Delta t N_n - \mu \Delta t N_n$$

on pose  $r = \lambda - \mu$

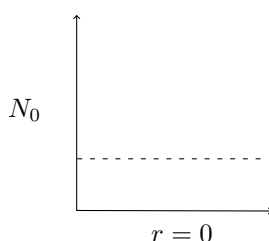
$$N_{n+1} = (1 + r \Delta t) N_n, \quad n = 0 \tag{1.1}$$

- Solution:  $N_n = (1 + r \Delta t)^n N_0, \quad n \in \mathbb{N}$

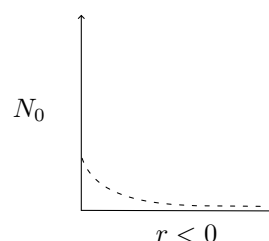
- Visualisation:  $\Delta t$  fixé



(a) Natalité supérieure à la mortalité



(b) Natalité égale à la mortalité



(c) Natalité inférieure à la mortalité

**Property.** .

- Lorsque  $t \rightarrow 0$ , la population semble tendre vers une courbe  $N(t) = N_0 e^{rt}$ , solution de  $\begin{cases} N'(t) = rN(t) \\ N(0) = N_0 \end{cases}$
- Si  $r > 0$ , la population croît indéfiniment
- Si  $r < 0$ , il y a extinction de l'espèce.

Inconvénients:

1. Une croissance infinie n'est pas réaliste
2. Pour être rigoureux, on devrait écrire  $E(rN_n)$  i.e partie entière.

## 1.2 Modèles continus

Motivation: L'observation qui prend  $\Delta t$  proche de 0 aura beaucoup plus d'information.

**Remark 1.1.** Le modèle de croissance géométrique

$$\begin{aligned} N(t + \Delta t) - N(t) &= \lambda \Delta t N(t) - \mu \Delta t N(t) \\ \Rightarrow \frac{N(t + \Delta t) - N(t)}{\Delta t} &= \lambda N(t) - \mu N(t) \end{aligned}$$

en faisant  $\Delta t \rightarrow 0$

$$N'(t) = \lambda N(t) - \mu N(t)$$

D'où l'équation des modèles continus:

$$\underbrace{N'(t)}_{\text{vitesse de variation}} = \underbrace{n(t)}_{\text{vitesse de naissance}} - \underbrace{m(t)}_{\text{vitesse de décès}} + \underbrace{i(t)}_{\text{vitesse d'immigration}} - \underbrace{e(t)}_{\text{vitesse d'émigration}}$$

### 1.2.1 Modèle de Malthus

- hypothèse:
  - solde migration nul:  $i(t) - e(t) = 0$
  - vitesse de naissance proportionnel à la population à l'instant  $t$ :  $n(t) = \lambda N(t)$
  - vitesse de décès:  $m(t) = \mu N(t)$
- Modèle:  $\begin{cases} N'(t) = (\lambda - \mu)N(t) \\ N(0) = N_0 \end{cases}$
- Solution:  $N(t) = N_0 e^{(\lambda - \mu)t}$
- **Property.** – Il peut être vu comme limite du modèle de croissance géométrique.
  - Lorsque  $r = \lambda - \mu > 0$  croissance est proportionnel.
  - Lorsque  $r = \lambda - \mu = 0$  la population n'évolue pas.
  - Lorsque  $r = \lambda - \mu < 0$  la population tend vers 0.
- Inconvénients:
  - croissance exponentielle pas réaliste. Il faut prendre en compte:
    - \* la limitation des ressources
    - \* l'interaction avec l'environnement

### 1.2.2 Modèle Verhulst

Corrige le modèle de Malthus en prenant en compte la limitation de ressources.

- Idée: limiter la croissance à un seuil  $K$  appelé capacité biotique

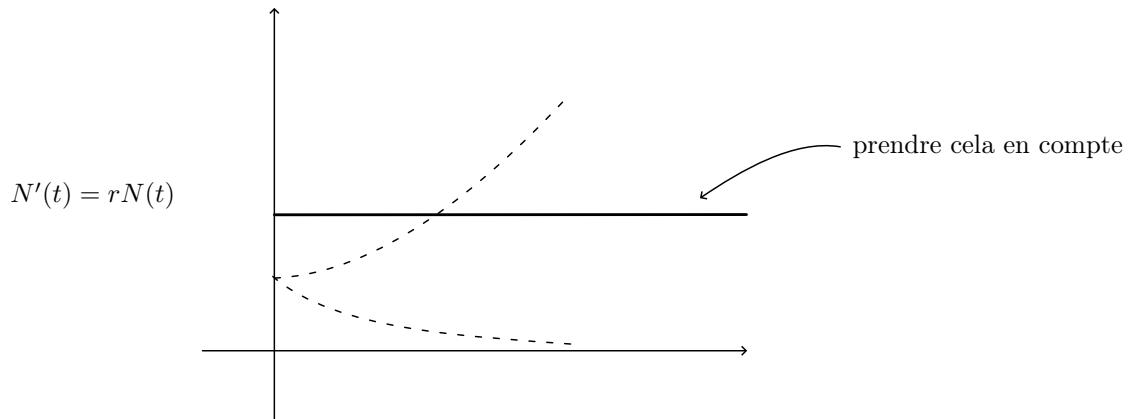


Figure 1.2: Modèle de Malthus



Figure 1.3: Modèle de Verhulst

- hypothèse: Sole de migration nul
  - taux de natalité fonction affine décroissante de la population  $\lambda \approx \lambda(1 - \frac{N(t)}{K})$
  - taux de mortalité fonction affine croissante de la population  $\mu \approx -\mu(1 - \frac{N(t)}{K})$
- Modèle:  $\begin{cases} N'(t) = rN(t)(1 - \frac{N(t)}{K}) \\ N(0) = N_0 \end{cases}$
- Solutions:  $N(t) = \frac{K}{1 + (\frac{K}{N_0} - 1)e^{-rt}}$   $t > 0$
- Visualisation:

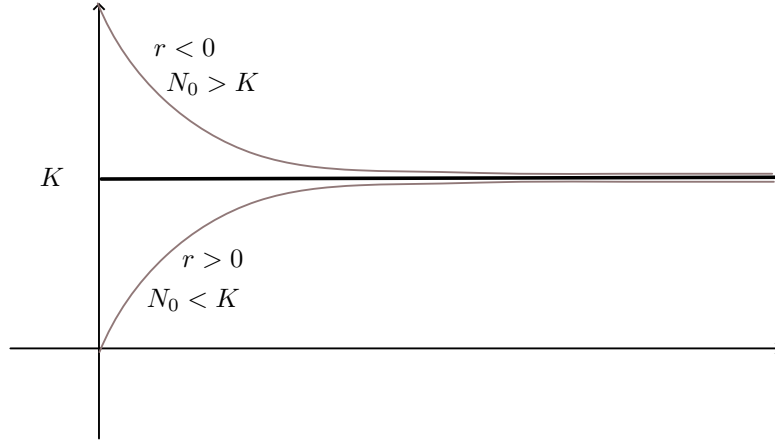


Figure 1.4: Verhulst solution

**Property.** Si  $r > 0$ , on a:

- si  $N_0 = 0$   $N_0 = K$  on a:  $N(t) = N_0 \forall t > 0$
- si  $0 < N_0 < K$ ,  $N$  croissante
- si  $N_0 > K$ ,  $N$  décroissante
- $N$  possède une limite si  $N_0 > 0$

$$\lim_{t \rightarrow \infty} N(t) = K$$

### 1.3 Modèle de croissance logistique

C'est un modèle discret

- hypothèse: i.e = 0  
 $n - m$  est une fonction affine de la population, i.e  $n - m = r\Delta t N(t)(1 - \frac{N(t)}{K})$
- Modèle: On suppose  $\Delta t = 1$ : On pose  $N_n = N(t_n)$

$$\text{On a: } \begin{cases} N_{n+1} - N_n = rN_n(1 - \frac{N_n}{K}) \\ N_0 \text{ donné} \end{cases}$$

**Property.** (À vérifier numériquement)

- si  $r < 2$ , la suite converge vers  $K$
- si  $2 < r < 2.449$ , la suite converge vers un cycle
- si  $2.449 < r < 2.57$ , la suite est encore un cycle mais plus complexe
- si  $r > 2.57$ , la suite devient chaotique

# Chapter 2

## cours 2

### 2.1 Notion de champ de vecteurs associée à une EDO

#### 2.1.1 Généralités et définitions

Les modèles continus de la dynamique de populations sont des problèmes de Cauchy pour les EDO.

$$(EDO) \begin{cases} y'(x) = f(t, y(t)) & t \in ]0, \pi[ \\ y(0) = y_0 \end{cases}$$

Où

$$\begin{aligned} y : [0, \pi] &\longrightarrow \mathbb{R} \\ t &\longmapsto y(t). \end{aligned}$$

$$\begin{aligned} f : ]0, \pi[ \times \mathbb{R} &\longrightarrow \mathbb{R} \\ (t, x) &\longmapsto f(t, x). \end{aligned}$$

- Si l'on sait résoudre analytiquement l'EDO (i.e donner l'expression de  $t \mapsto y(t)$ ) alors c'est terminé car il suffit d'étudier la fonction  $t \mapsto y(t)$
- Si l'on ne sait pas déterminer la solution analytique, on peut:
  1. s'assurer de **l'existence** et **l'unicité** de la solution et de sa **stabilité** vis à vis des données du problème.
  2. Puis analyser les propriétés qualitatives de cette solution pour simple analyse de  $f(t, x)$

**C'est ici qu'intervient les champs de vecteurs.**

Illustrations.

1. Prenons le modèle de Malthus

$$\begin{cases} N'(t) = rN(t), & t \in ]0, \pi[ \\ N(0) = N_0 \end{cases}$$

On sait que  $N(t) = N_0 e^{rt}$

2. Voici ce que fait python pour traiter  $N$ .

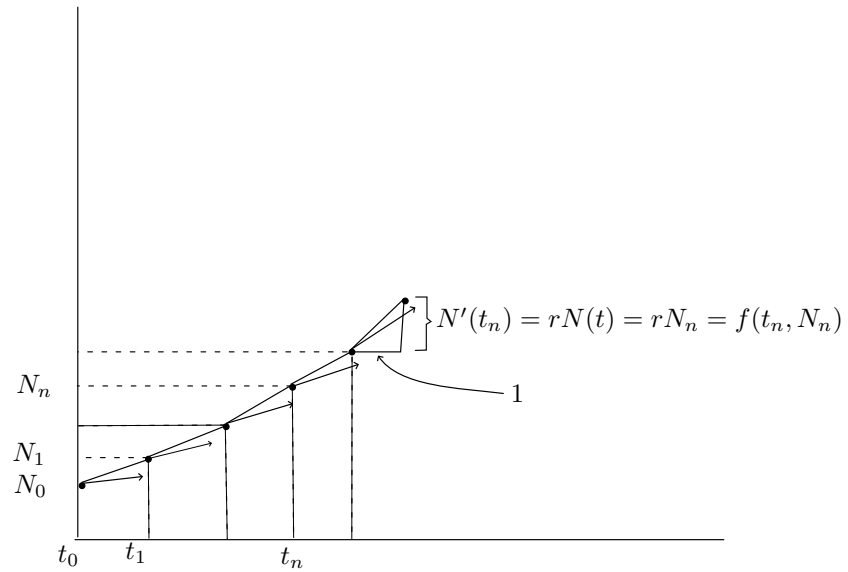


Figure 2.1: Ce que fait python

3. Traitons les vecteurs tangents à la courbe  $t \mapsto N(t)$  aux points  $t_n$ ,  $n = 0$
4. Si l'on connaît les valeurs minimales et maximales de la solutions on peut avoir l'allure de la solution.

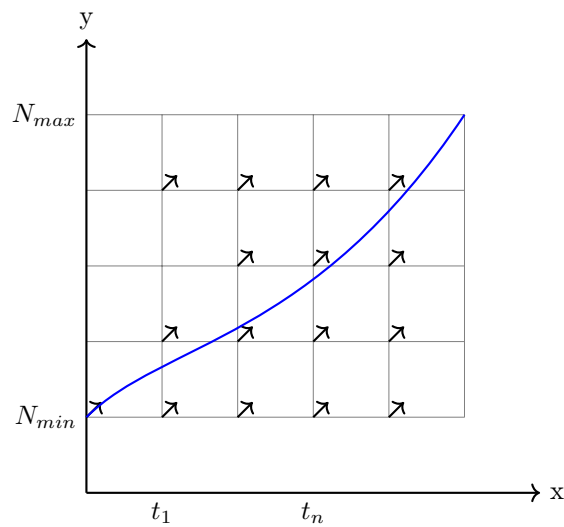


Figure 2.2: Une courbe sur des champs de vecteurs

Analysons ce que représente le vecteurs tangent:

- pour une courbe  $y = g(x)$
- python et tout autre logiciel procède ainsi



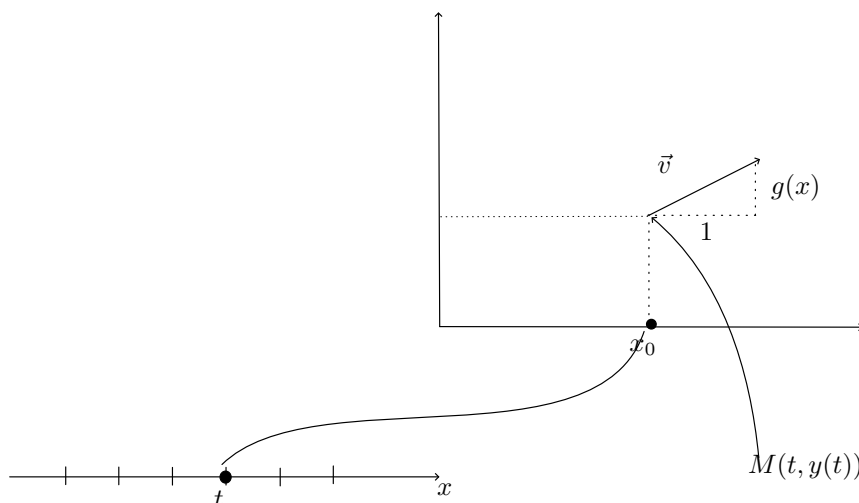


Figure 2.3: Ce que represente vecteur

Le vecteur tangent à la courbe:

$$\begin{aligned}\vec{v} &= (1, g'(x)) = (1, \frac{dy}{dx}) = (1, \frac{\frac{dy}{dt}}{\frac{dx}{dt}}) \\ &= \frac{1}{\frac{dx}{dt}} (\frac{dx}{dt}, \frac{dy}{dt}) = \frac{1}{\dot{x}(t)} \underbrace{(\dot{x}(t), \dot{y}(t))}_{\substack{\in \mathbb{R} \\ \text{vecteur tangent}}} \\ \vec{v} &= (\dot{x}(t), \dot{y}(t))\end{aligned}$$

Càd  $\vec{v}$  est le vecteur vitesse au points  $M(x(t), y(t))$  a la courbe paramétrée  $t \mapsto \begin{cases} x(t) = t \\ y(t) = g(t) \end{cases}$ . On a le résultat.

### Proposition 2.1.

(y obtient solution de l'EDO  $y'(t) = f(t, y(t))$ )

$\Updownarrow$

(vecteur vitesse de la courbe paramétrée  $t \mapsto (x(t), y(t))$  au point  $M(t_0) = (t_0, y(t_0))$  si le vecteur  $(1, f(t_0, y(t_0)))$ )

### Proposition 2.2.

$$\begin{aligned}V : \mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (t, y) &\longmapsto V((t, y)).\end{aligned}$$

(si le champ de vecteur associé à l'EDO  $y'(t) = f(t, y(t)) \Leftrightarrow V(t, y) = (1, f(t, y))$ )

## 2.1.2 Dessins de champs de vecteurs

### Principe:

À chaque points  $P = (p_x, p_y)$  on trace le vecteur  $\varepsilon V(P)$  où  $\varepsilon$  est une constance positive choisi pour écrire les vecteurs trop longs.

Avec python on écrit `quiver(Px, Py, Vx, Vy, angles='xy')` RQ 1: Cette fonction est vectorielle, i.e  $P_x, P_y, V_x, V_y$ , sont des numpy array de taille  $n$ . RQ 2: On peut ajouter un paramètre pour contrôler la longueur des vecteurs:

`plt.quiver(Px, Py, Vx, Vy, angles='xy', scale=1)`

Par conséquent, il faut normaliser les vecteurs (i.e le champ de vecteur)

**Example 2.3.** Champ de vecteur du modèle de Verhulst:

```
def f(t, y):
    return r * y * (1 - y/k)
```

la grille:

```
lt = np.linspace(tmin, tmax, N+1)
ly = np.linspace(ymin, ymax, M+1)
T, Y = np.meshgrid(lt, ly)
```

Construire les vecteurs:

```
Y = 1 + 0 * T
V = f(T, Y)
norm = np.sqrt(U*U + V*V)
U = U/norm
V = V/norm
```

On place les points:

```
plt.scatter(T, Y, marker='+', alpha = 0.5)
```

On place les vecteurs

```
plt.quiver(T, Y, U, V, angles='xy', scale=N)
```

### 2.1.3 Recherche de solution approchée de modèles sous python

On cherche une solution approchée de

$$\begin{cases} y'(t) = f(t, y(t)) & t \in ]t_0, t_0 + T[ \\ y(t_0) = y_0 \end{cases}$$

avec python. Pour cela il suffit de dire **en quels points** on veut cette solution.

On se donne:

- une liste des instants  $[t_0, t_1, \dots, t_N]$
- $t_0, y_0$
- Puis, on appelle la fonction `odeint` du module `scipy.integrate` de python.
- On obtient une liste  $[y_0, y_1, \dots, y_N]$

**Example 2.4.** Cas du modèle du Verhulst

- EDO:

```
def f(t, y):
    return \ldots
```

- Instants

```
t0, tf = a, b
N = 100
t = np.linspace(t0, tf, N)
```

- On appelle odeint

```

1 from scipy.integrate import odeint
2 yapp = odeint(f, t, y), rtol=None, atol=None, tfloat=False)
3 plt.plot(t, yapp, \ldots)

```

## 2.2 Modèle de prédateur proie (lotka-voltena (1931))

$H(t)$ : population de sardins

$P(t)$ : population de requins

$$\frac{H'(t)}{H(t)} = \text{taux de variation de sardins} = \underbrace{a}_{\text{taux de croissance}} - \underbrace{bP(t)}_{\text{taux de mortalité}}$$

$$\frac{P'(t)}{P(t)} = \text{taux d'arrivé des requetes} = \underbrace{-c}_{\text{taux de décès}} + \underbrace{dH(t)}_{\text{taux de croissance}}$$

D'où le modèle:

$$\begin{cases} H'(t) = H(t)(a - bP(t)) & t > 0 \\ P'(t) = P(t)(-c + dH(t)) \\ H(0) = H_0, \quad P(0) = P_0 \end{cases}$$

Si l'on désigne par  $p \geq 0$  la proportion des requêtes en sardines pêchés

$$\begin{cases} H'(t) = H(t)(a - p - bP(t)) & t > 0 \\ P'(t) = P(t)(-c - p - dH(t)) \\ H(0) = H_0 \\ P(0) = P_0 \end{cases}$$

# Chapter 3

## cours3: Interpolation polynomiale

On va essayer de construire des polynômes qui passent par un ensemble (nuages) de points donnés.  
Si ces points sont les valeurs d'une fonction, on aimerait:

- savoir si le polynôme construit est d'autant plus proche de la fonction que le nombre de point est grand. C'est-à-dire, est-ce que nute des "erreurs" tend vers zero lorsque le nombre de points tend vers l'infini.
- Si oui, comment quantifier cette convergence? C'est-à-dire, quelle est la vitesse (ordre) de cette convergence.

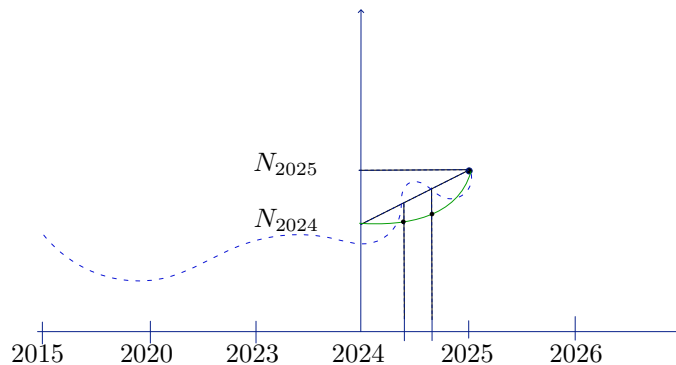


Figure 3.1: evolution-de-population-en-annee

1. Approche 1: approximation linéaire.
  - Polynôme de degré 1
2. Approche 2:
  - polynôme de degré 1
  - approximation quadratique
3. Approche 3: prise en compte d'Historique

## Rappels sur les nuts numériques

### Vitesse (ordre) de convergence

### valeur ajoutée par itérations

**Definition 3.1.** Soit  $(x_n)_n \subset \mathbb{R}^n$  une suite qui converge vers  $x^* \in \mathbb{R}^n$ , pour une norme  $\|\cdot\|$  de  $\mathbb{R}^n$

- Si  $k_1 = \lim_{n \rightarrow \infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|}$  existe et  $k_1 \in ]-1, 1[ \setminus \{0\}$ . On dit que la suite converge linéairement vers  $x^*$  ou que la convergence est d'ordre 1.
- Si  $k_1 = 0$ ,  $k_2 = \lim_{n \rightarrow \infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^2}$  existe et non nul. On dit que la suite converge quadratiquement vers  $x^*$ , ou que la convergence est d'ordre 2.
- Si  $k_q = \lim_{n \rightarrow \infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q}$  existe et  $\neq 0$  la convergence est d'ordre  $q$ . La constante  $K_q$  est appelée constante asymptotique d'erreur.

**Example 3.2.** 1.  $x_n = (0.2)^n$

- On a  $\lim_{n \rightarrow \infty} x_n = 0$ . La convergence vers  $x^* = 0$ .
- $\lim_{n \rightarrow \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|} = \lim_{n \rightarrow \infty} \frac{(0.2)^{n+1}}{(0.2)^n} = 0.2 \in ]-1, 1[ \setminus \{0\}$

D'où

- $x_n$  converge à l'ordre 1
- Sa constante asymptotique est  $k_1 = 0.2$

2.  $I_n = (0.2)^{2^n}$ . On a  $\lim_{n \rightarrow \infty} I_n = 0$

On a:

$$\begin{aligned} I_{n+1} &= (0.2)^{2^{n+1}} = (0.2)^{2^n \cdot 2} \\ &= \left( (0.2)^{2^n} \right)^2 \\ &= (I_n)^2 \end{aligned}$$

D'où  $\lim_{n \rightarrow \infty} \frac{I_{n+1}}{(I_n)^2} = \lim_{n \rightarrow \infty} \frac{(I_n)^2}{(I_n)^2} = 1$  D'où

- convergence d'ordre 2
- de constante  $k_2 = 1$

En pratique, on ne dispose pas de  $K_q$

**Definition 3.3.**

$$x_n \text{ converge vers } x^* \text{ à l'ordre } q \Rightarrow \exists N \in \mathbb{N}, \exists A, B \in \mathbb{R} \text{ tq } \forall n \geq N, 0 < A \leq \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} \leq B < +\infty$$

La convergence est au moins d'ordre  $q$  si et seulement si on a (deuxieme partie d'équation)

### 3.0.1 Valeur ajoutée par l'itération

Il est question de comparer 2 suites qui ont la même vitesse de convergence.

**Remark 3.4.** Si  $|x_n - x^*| = 4 \cdot 10^{-8} = 0.\underbrace{0000000}_7 \text{ chiffres} 4$ . On dira que  $x_n$  et  $x^*$  ont 7 chiffres exactes apres la

virgule.

$$\begin{aligned}\log_{10} |x_n - x^*| &= \log_{10} 4 - 8 \log_{10}(10) \\ \frac{\log |x_n - x^*|}{\log 10} &= \frac{\log 4}{\log 10} - 8\end{aligned}$$

i.e  $d_n = -\log_{10} |x_n - x^*|$  mesure de nombre de chiffres décimales entre  $x_n$  et  $x^*$  qui coïncident.

**Remark 3.5.**

$$\lim_{n \rightarrow \infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} = K_q \Rightarrow K_q \approx \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q}$$

D'où  $d_{n+1} - qd_n \approx -\log_{10} K_q$ , i.e

$$d_{n+1} + \frac{\log_{10} K_q}{1-q} \approx q(d_n + \frac{\log_{10} K_q}{1-q})$$

Donc, le nombre de chiffres significatives est multiplié par  $q$ .

**Proposition 3.6.** Si  $x_n$  converge à l'ordre 1 vers  $x^*$  de constante asymptotique  $K_1$ , alors le nombre d'itérations nécessaires pour gagner un chiffre exacte est la partie entière de  $-\frac{1}{\log_{10} K_1}$

**Proof.** Soit  $m$  le nombre d'itérations pour gagner un chiffre. Comme  $d_{n+m} - d_n = -\log_{10} K_1$ , en partant de  $d_n$ , après  $m$  itérations on aura

$$d_{n+m} - d_n = -m \log_{10} K_1$$

D'où on aura gagné 1 chiffre si  $d_{n+m} - d_n = 1$ , i.e

$$1 = -m \log_{10} K_1 \Rightarrow m = \left( -\frac{1}{\log_{10} K_1} \right)$$

□

### 3.0.2 Obtenir numériquement la vitesse de convergence

On cherche  $q$  tq:  $\lim_{n \rightarrow \infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} = K_q \in \mathbb{R}^*$

**Remark 3.7.**

$$\frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} \approx K_q \Rightarrow \underbrace{\log \|x_{n+1} - x^*\|}_Y - \underbrace{q \log \|x_n - x^*\|}_X = \log K_q$$

i.e  $Y = aX + b$ .

Conclusion: pour déterminer  $q$ :

- Traiter la courbe  $\log \|x_n - x^*\| \mapsto \log \|x_{n+1} - x^*\|$
- Déterminer  $q$  comme la pente de la droite passant par le maximum de points.

$$x_n = x_0, x_1, \dots, x_N$$

$$x_n - x^* = x_0 - x^*, x_1 - x^*, \dots, x_N - x^*$$

$$x_{n+1} - x^* = x_1 - x^*, x_2 - x^*, \dots, x_{N+1} - x^*$$

En python:

```
1 xn = np.array([x0, ..., xN])
2 e = np.log(np.abs(xn - x*))
```

```

3 | ex = e[0:-1] #de premier a avant dernier
4 | ey = e[1:] #de deuxieme au dernier
5 | plt.scatter(ex, ey, label="miage")
6 | a,b = np.polyfit(ex, ey, 1)
7 | plt.plot(ex, b + a * ex, label=f"$x \mapsto {b:32f} + {a:32f}x$")

```