Analyse numérique avec python

Yehor Korotenko

February 4, 2025

Contents

Chapter 1

Équations Différentielles

1.1 Modèles discrètes

On diésigne par $N(t)$ la population d'individus à l'instant t.
Équation du modèle discret:

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

1.1.1 Modèle de croissance géomètrique

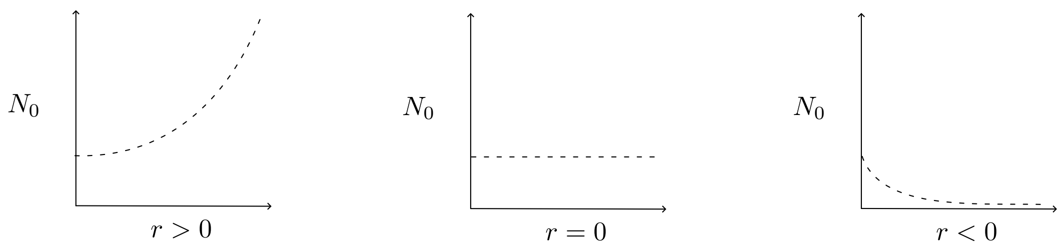
- hypothèse:
  - solde migration nul: i.e $i - e = 0$
  - nombre de croissance proportionnel à la taille de la population $\underbrace{n = \lambda \Delta t N(t)}_{\text{taux de natalité}}$
  - Idem pour le mobre de décès: $\underline{m = \mu \Delta t N(t)}_{\text{taux de mortalité}}$
- Modèle: On pose $N_n = N(t_n)$ 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 $r = \lambda - \mu$

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

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



(a) Natalité supérieure à la mortalité

(b) Natalité égale à la mortalité

(c) Natalité inférieure à la mortalité

Property. .

- Lorsque $t \to 0$, 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'éspèce.

Inconvenients:

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 continues

Motivation: L'observation qui prend Δt proche de 0 aura beaucoup plus d'information.

> **Remark 1.1.** Le modèle de croissance géomètrique
>
> $$\begin{aligned} N(t+\Delta t) - N(t) &= \lambda \Delta t N(t) - \mu \Delta t N(t) \\ \Rightarrow \frac{N(t+\Delta t) - N(t)}{\Delta t} &= \lambda N(t) - \mu N(t) \end{aligned}$$
>
> en faisant $\Delta t \to 0$
>
> $$N'(t) = \lambda N(t) - \mu N(t)$$
>
> D'où l'équation des modèles continues:
>
> $$\underbrace{N'(t)}_{\text{vitesse de variation}} = \underbrace{n(t)}_{\text{vitesse de naissance}} - \underbrace{m(t)}_{\text{vitesse de décès}} + \underbrace{i(t)}_{\text{vitesse d'immigration}} - \underbrace{e(t)}_{\text{vitesse d'émigration}}$$

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 si comme limite du modèle de croissance géomètrique.
  - 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.
- Inconvenients:
  - 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 prennant en compte la limitation de ressources.

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

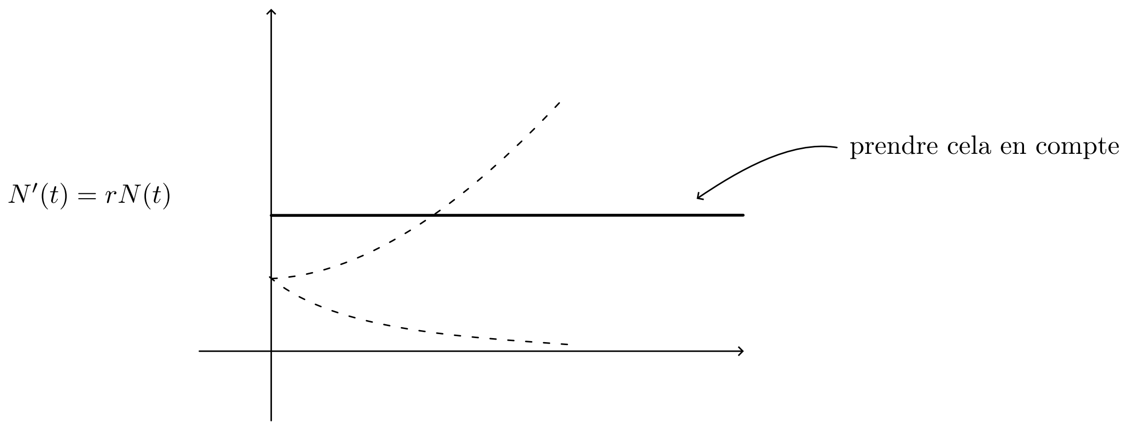


Figure 1.2: Modèle de Malthus

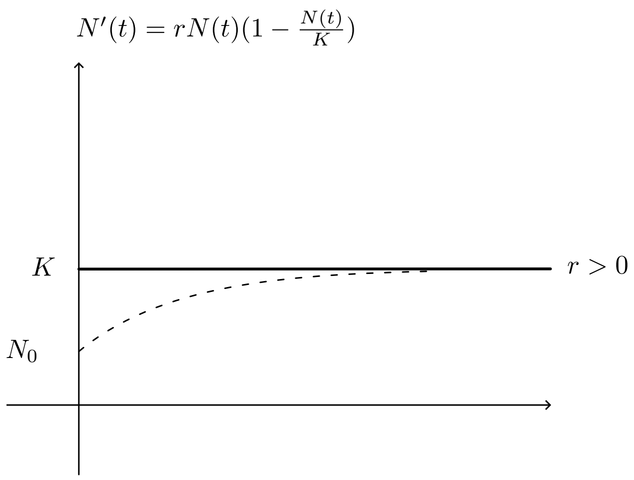


Figure 1.3: Modèle de Verhulst

- hypothèse: Sole de migration nul
  - taux de natalité fonction afiine 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: $N(t) = \frac{K}{1+(\frac{K}{N_0}-1)e^{-rt}} \quad t > 0$
- Visualisation:

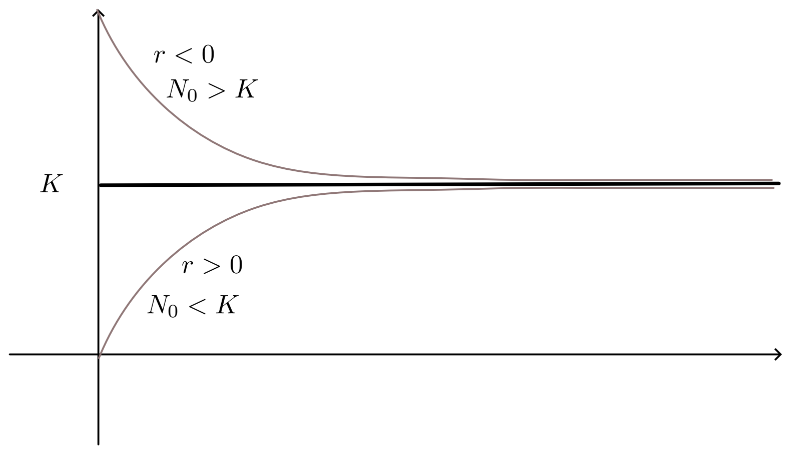


Figure 1.4: Verhulst solution

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 $N_0 > 0$

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

1.3 Modèle de croissance logistique

C'est un modèle discrét

- 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 $N_n = N(t_n)$

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

Property. (À vérifier numeriquement)

- 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 complèxe
- si $r > 2.57$, la suite devient chaotique

1.4 Notion de champ de vecteurs associée à une EDO

1.4.1 Généralités et définitions

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

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

Où

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

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

- Si l'on sait résoudre analytiquement l'EDO (i.e donner l'expression de $t \mapsto y(y)$) alors c'est terminé car il suffit d'étudier la fonction $t \mapsto y(t)$
- Si l'on ne sait pas détérminer la solution analytique, on peut:
  1. s'assurer de **l'éxistence** 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.

Illustations.

1. Prenons le modèle de Malthus
$$\begin{cases} N'(t) = rN(t), \quad t \in]0,\pi[\\ N(0) = N_0 \end{cases}$$
On sait que $N(t) = N_0 e^{rt}$
2. Voici ce que fait python pour traiter N.

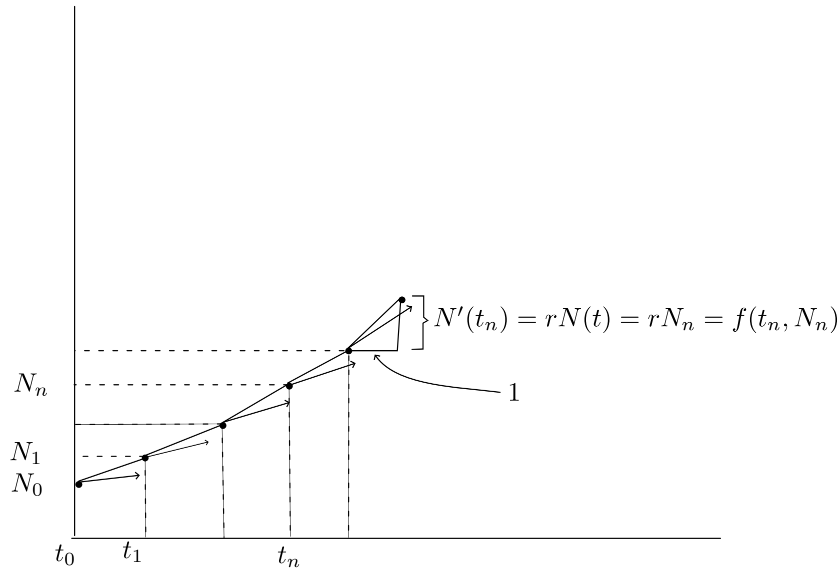


Figure 1.5: Ce que fait python

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 minimals et maximales de la solutions on peut avoir l'allure de la solution.

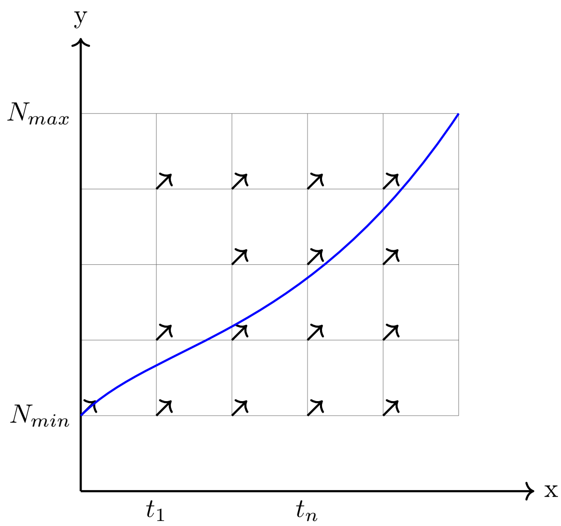


Figure 1.6: Une courbe sur des champs de vecteurs

Analysons ce que represente le vecteurs tangent:

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

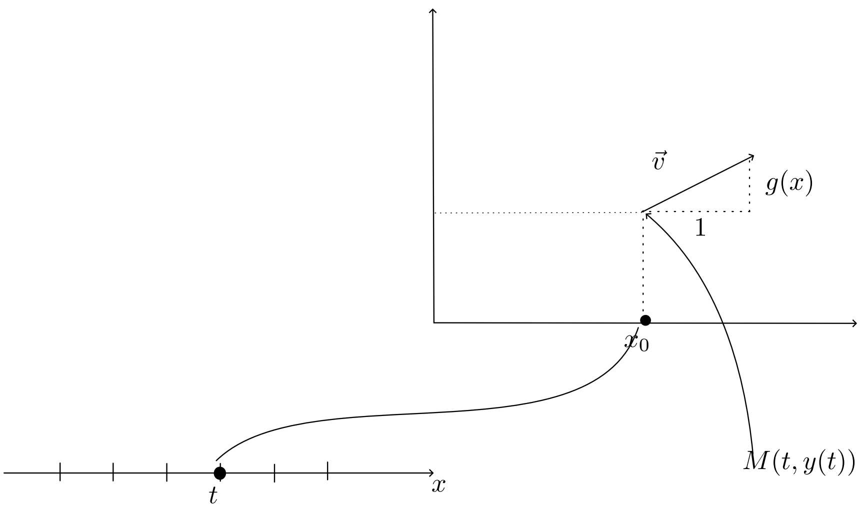


Figure 1.7: Ce que represente vecteur

Le vecteur tangent à la courbe:

$$\vec{v} = (1, g'(x)) = (1, \frac{dy}{dx}) = (1, \frac{\frac{dy}{dt}}{\frac{dy}{dt}})$$

$$= \frac{1}{\frac{dy}{dt}}(\frac{dx}{dt}, \frac{dy}{dt}) = \underset{\in\mathbb{R}}{\frac{1}{\dot{x}(t)}} \underbrace{(\dot{x}(t), \dot{y}(t))}_{\text{vecteur tangent}}$$

$$\vec{v} = (\dot{x}(t), \dot{y}(t))$$

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

Proposition 1.2.

$$(\text{y obtient solution de l'EDO } y'(t) = f(t, y(t)))$$
$$\Updownarrow$$
$$(\text{vecteur vitesse de la courbe parametrée } t \mapsto (x(t), y(t)) \text{ au point } M(t_0) = (t_0, y(t_0))$$
$$\text{si le vecteur } (1, f(t_0, y(t_0))))$$

Proposition 1.3.

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

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

1.4.2 Dessins de champs de vecteurs

Principe:

À chaque points $P = (p_x, p_y)$ on trace le vecteur $\varepsilon V(P)$ où ε est une constance positive choisi pour écrire les vecteurs trop longs.
Avec python on écrit *quiver(P_x, P_y, V_x, V_y, 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 controles la longeur des vecteurs:

$$\text{plt.quiver}(P_x, P_y, V_x, V_y, \text{ angles='xy', sacle=1})$$

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

Example 1.4. 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(lx, 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)
```

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

Example 1.5. 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

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

1.5 Modèle de prédateur prose (lotka-voltena (1931))

$H(t)$: population de sardins
$P(t)$: pupulation de reguins

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

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

D'où le modèle:

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

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

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

Chapter 2

Interpolation polynomiale

On va essayer de construire des polynôms qui passent par un ensemble (nuages) de points donnés.

Si ces points sont les valeurs d'une fonction, on amerait:

- savoir si le polynôme construit est d'autant plus proche de la fonction que le nombre de point est grand. C'est-à-dre, 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.

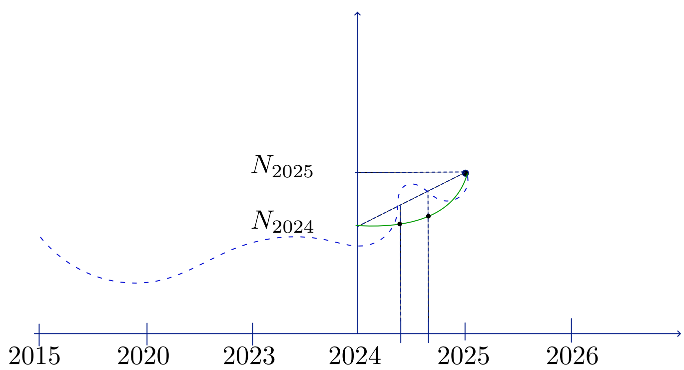


Figure 2.1: evolution-de-population-en-annee

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

2.1 Rappels sur les nuts numériques Vitesse (ordre) de convergence valeur ajoutée par itérations

Definition 2.1. Soit $(x_n)_n \subset \mathbb{R}^n$ une suite qui converge vers $x^* \in \mathbb{R}^n$, pour une norme $\| \ \|$ de $\mathbb{R}^n$

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

Example 2.2.

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

   D'où

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

2. $I_n = (0.2)^{2^n}$. On a $\lim_{n\to\infty} I_n = 0$
   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 $\lim_{n\to\infty} \frac{I_{n+1}}{(I_n)^2} = \lim_{n\to\infty} \frac{(I_n)^2}{(I_n)^2} = 1$ D'où

   - convergence <u>d'ordre 2</u>
   - de constante $k_2 = 1$

En pratique, on ne dispose pas de K_q

Definition 2.3.

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

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

2.1.1 Valeur ajoutée par l'itération

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

Remark 2.4. 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.

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

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

Remark 2.5.

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

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 2.6. 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 partié enitère de $-\frac{1}{\log_{10} K_1}$

Proof. Soit m le nombre d'itérations pour gegner un chiffre. Comme $d_{n+1} - d_n = -\log_{10} K_1$, 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

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

□

2.1.2 Obtenir numériquement la vitesse de convergence

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

Remark 2.7.

$$\frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} \approx K_q \Rightarrow \qquad \underbrace{\log\|x_{n+1} - x^*\|}_{Y} - q\underbrace{\log\|x_n - x^*\|}_{X} = \log K_q$$

i.e $Y = aX + b$.

Conclusion: pour détérminer q:

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

$$x_n = x_0, x_1, \ldots, x_N$$
$$x_n - x^* = x_0 - x^*, x_1 - x^*, \ldots, x_N - x^*$$
$$x_{n+1} - x^* = x_1 - x^*, x_2 - x^*, \ldots, x_{N+1} - x^*$$

En python:

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

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

2.2 Interpolation: définition-motivation-exemples

2.2.1 Définition

Definition 2.8. Soient $(x_i, y_i)_{i=\{1,\dots,N\}}$ un nuage de points(exemple un ensemble discret de point du graphe d'une fonction). Interpoler ce nuage de points correspond à chercher un polynôme de degré $N-1$, qui passe par chacun de ces points.

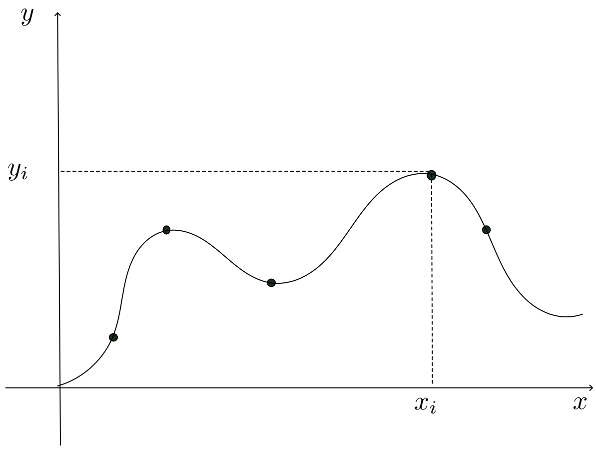


Figure 2.2: L'exemple visuel de la définition

Questions:

1. Comment le construire?
2. $P_{N-1} \in \mathbb{R}_{N-1}[X]$
3. $P_{N-1}(x_i) = y_i$

2.2.2 Motivations

- La solution d'un problème est fournie par une formule représentative: Noyau de la chaleur (i.e convolution) est un cherche la solution en un nombre de points.
  - On approche alors la fonction par un polynôme: i.e chercher le polynôme de degré "bas" proche de la fonction
- La solution d'un problème n'est connue qu'à table des valeurs en un nombre fini de points et on souhaite l'évaluer partout.
  - l'intérpolation
- On peut utiliser l'intérpolation dans
  - l'intégration numérique
  - la résolution numérique des EDO
  - la visualisation scientifique

Definition 2.9. Un tel polynôme est appelé **polynôme intérpolateur de lagrange** de degré $N-1$ de ces points.

2.2.3 exemples d'intérpolation

Theorem 2.10. Polynôme intérpolateur de degré 1.
Soient (x_1, y_1), (x_2, y_2) 2 points distincts de $\mathbb{R}^2$

- Il existe une unique droite passant par les 2 points.

$$(x,y) \in \mathcal{D} \Leftrightarrow (x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)=0$$

- Si de plus, $x_1 \neq x_2$, il existe un unique polynôme de degré 1 (i.e $P_1 \in \mathbb{R}_1[X]$) tq:

$$(x,y) \in \mathcal{D} \Leftrightarrow y = P(x) \text{ avec } P_1 = \frac{(x-x_1)y_1-(x-x_2)y_2}{x_2-x_1}$$

Proof.

- $$\begin{aligned} M\begin{pmatrix} x \\ y \end{pmatrix} \in \mathcal{D} &\Leftrightarrow M\vec{M_1}/M_1\vec{M_2} \\ &\Leftrightarrow det(M\vec{M_1}, M_1\vec{M_2}) = 0 \\ &\Leftrightarrow \begin{vmatrix} x-x_1 & x_2-x_1 \\ y-y_1 & y_2-y_1 \end{vmatrix} = 0 \\ &\Leftrightarrow (x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)=0 \end{aligned}$$

- Si $x_1 \neq x_2$

$$\begin{aligned} M \in \mathcal{D} &\Leftrightarrow y - y_1 = \frac{(x-x_1)(y_2-y_1)}{x_2-x_1} \\ &\Leftrightarrow y = P_1(X) \end{aligned}$$

□

Remark 2.11. On a l'écriture équivalente de P_1:

- $$P_1 \frac{x_0y_1 - x_1y_2}{x_2-x_1} + X\frac{y_2-y_1}{x_2-x_1} \equiv a_0 + a_1X$$

  C'est l'écriture dans la base $(1, X)$ de $\mathbb{R}_1[X]$

- $$P_1 = \underbrace{\frac{x-x_2}{x_1-x_2}}_{l_1} y_1 + \underbrace{\frac{x-x_1}{x_2-x_1}}_{l_2} y_2$$

  C'est l'écriture dans la base (l_1, l_2) de $\mathbb{R}_1[X]$
  RQ:

  $$l_1(x_1) = 1 \quad l_1(x_2) = 0 \quad l_2(x_1) = 0 \quad l_2(x_2) = 1$$

  (base de lagrange)

- $$P_1 = y_1 + \frac{y_2-y_1}{x_2-x_1}(x-x_1)$$

  C'est l'écriture dans la base $(1, x-x_1)$ de $\mathbb{R}_1[X]$ (base de newton)

Example 2.12. Méthode de calcul employle
Chercher le polynôme interpolateur de lagrange aux points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$

- Méthode 1: $x_1 \neq x_2 \neq x_3$
  P_2 est un polynôme de degré 2
$$P_2 = a_0 + a_1 x + a_2 x^2$$

Lemma 2.13.
$$P_2(x_1) = y_1, \quad P_2(x_2) = y_2 \qquad \text{i.e } a_0 + a_1 x_1 + a_2 x_1^2 = y_1$$

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}} \Rightarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \underbrace{M^{-1}}_{???} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Remark 2.14. Par 2 points
$$M = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \text{ et } M^{-1} = \frac{1}{x_2 - x_1} \begin{bmatrix} x_2 & -x_1 \\ -1 & 1 \end{bmatrix}$$

- Méthode 2:

$$P_2 = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$$

$$\underset{i=1,2,3}{P_2(x_i) = y_i} \Rightarrow \begin{cases} a_0 & = y_1 \\ a_0 + a_1(x_2 - x_1) & = y_2 \\ a_0 + a_1(x_3 - x_1) + a_2(x_3 - x_1)(x_3 - x_2) & = y_3 \end{cases}$$
$$\Rightarrow \begin{cases} a_0 & = y_1 \\ a_1 & = \frac{y_2 - y_1}{x_2 - x_1} \\ a_2 & = \frac{y_3 - y_1 - \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_1)}{(x_3 - x_1)(x_3 - x_2)} \end{cases}$$

Remark 2.15. On a:
$$a_2 = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_3 - y_2}{x_3 - x_2}}{x_3 - x_1}$$
càd
$$P_3 = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_3 - y_2}{x_3 - x_2}}{x_3 - x_2}(x - x_1)(x - x_2)$$

x_1	$y_1 =: a_3$		
x_2	y_2	$\frac{y_2 - y_1}{x_2 - x_1} =: a_1$	
x_3	y_3	$\frac{y_3 - y_2}{x_3 - x_2}$	$\frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_3 - y_2}{x_3 - x_2}}{x_3 - x_1} =: a_2$

- Méthode 3:

$$P_3 = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}y_3 = \sum_{i=1}^{3} \underbrace{\left(\prod_{j=1}^{3} \frac{x-x_j}{x_i-x_j}\right)}_{l_i(x)} y_i$$

Remark 2.16. $(x_1, y_1), (x_2, y_2)$

$$y = \frac{x-x_2}{x_1-x_2}y_1 + \frac{x-x_1}{x_2-x_1}y_2$$

2.3 Polynôme interpolateur de lagrange

2.3.1 Définition et propriétés

Theorem 2.17. (existence et utilité)
Soit $x_1, \ldots, x_n$ des réels 2 à 2 distincts et $y_1, \ldots, y_n$ des réels quelconques: Il existe un unique polynôme $P \in \mathbb{R}_{n-1}[X]$ (i.e de degré $n-1$) tel que $p(x_i) = y_i$, $i = 1, \ldots, n$
On dit que P est le polynôme interpolateur de lagrange aux points (x_1, y_1), ..., (x_n, y_n)

Proof. Soit

$$\begin{aligned} \Phi : \mathbb{R}_{n-1}[X] &\longrightarrow \mathbb{R}^{n-1} \\ P &\longmapsto \Phi(P) = (P(x_1), \ldots, P(x_n)). \end{aligned}$$

on a:

- Φ linéaire
- Φ injective

En effet, $\Phi(P) = 0 \Leftrightarrow P(x_i) = 0 \Leftrightarrow P \equiv 0$ car $deg(P) \leq n-1$. D'où Φ isomorphisme d'espace vectoriel et la surjection assure le résultat. □

Definition 2.18. Si f est continue sur $[a, b] \to \mathbb{R}$, $x_1, \ldots, x_n \in [a, b]$ 2 à 2 distincts, alors, l'unique $P \in \mathbb{R}_{n-1}[X]$ tq $P(x_i) = f(x_i)\, i = 1, \ldots, n$ est appelé polynôme d'interpolation de lagrange de f aux points $x_1, \ldots, x_n$

2.3.2 Estiomation d'erreur

Theorem 2.19. l'erreur
Soient

- $a < b$ $f : [a, b] \to \mathbb{R}$ continue
- $x_1, \ldots, x_n$ 2 à 2 distincts de $[a, b]$
- P polyôme d'interpolation de lagrange de f aux points x_i

Si f est n fois dérivable sur $]a, b[$, alors, pour tout $a \in [a, b]$, il existe $t \in]a, b[$ tq

$$f(x) - P(x) = \omega_n(x) \frac{f^{(n)}(t)}{n!}$$

où $\omega_n(x) = (x - x_1) \dots (x - x_n)$

Corollary 2.20. Si $f^{(n)}$ est bornée par M sur $]a, b[$, alors $\forall x \in [a, b]$

$$|f(x) - P(x)| \leq \frac{M}{n!}|\omega_n(x)| \leq \frac{M}{n!}(b-a)^n$$

Proof. à faire □

2.3.3 Implémentation avec python

```
from scipy.interpolite import lagrange
x = np.array([x_1, x_2, x_3])
y = np.array([y_1, y_2, y_3])
p = lagrange(x, y)
print(p) # affiche le polynome
print(p(3)) # collable
```

2.4 Construction des polynôme d'intérpolation de lagrange

$x_0, \dots, x_{n-1}$ 2 à 2 distincts

2.4.1 Interpolation dans la base canonique (Vandermonde)

Construction

$$P = \sum_{i=1}^{n-1} a_i x^i$$

$$P(x_i) = y_i,\ i = 0, \dots, n-1$$

$$\underbrace{\begin{bmatrix} 1 & x_0 & \dots & x_0^{n-1} \\ 1 & x_1 & \dots & x_1^{n-1} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & \dots & x_{n-1}^{n-1} \end{bmatrix}}_{V(x_0,\dots,x_{n-1})} \underbrace{\begin{bmatrix} a_0 \\ \dots \\ a_{n-1} \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} y_1 \\ \dots \\ y_{n-1} \end{bmatrix}}_{b}$$

Matrice de Vandermonde

- elle pleine
- malconditionnée

```
def VDM_Mat(x):
    x_n = np.reshape(x, (x.size, 1))
    return x_n ** np.arange(x.size)
```

```
def VDM_Poly(x, y):
    M = VDM_Mat(x)
    a = np.linalg.solve(M, y)
    return a
```

Evaluation efficace de P algorithme de Horner

Proposition 2.21. Si X est un réel et Q est le polynôme défini par

$$Q(X) = a_0 X^n + a_1 X^{n-1} + \dots + a_{n-1} X + a_n$$

alors la suite

$$\begin{cases} q_0 = a_0 \\ q_k = q_{k-1}x + a_k,\ k = 1, \dots, n \end{cases}$$

vérfiie $q_n = Q(x)$

Proof. (laissé exo)

$$Q(X) = X^2 + 2X + 1 \equiv (X+2)X + 1$$

□

```
def Horner(P, xx):
   y = 0
   for a in P:
        y = y * xx + a
   return y
```

```
def IntuP_VDM(x, y, xx):
    a = VDM_Poly(x, y)
    yy = Horner(a[::-1], xx)
    return yy
```

2.4.2 Interpolation dans la base duale: Formule de lagrange et points barycentrique

Construction

Idée prendre pour base de $\mathbb{R}_{n-1}[X]$ l'image réciproque de la base canonique de $\mathbb{R}^{n-1}$ pour Φ

$$L_i(x_j) = \begin{cases} 1 \text{ si } i \neq j \\ 0 \text{ sinon} \end{cases}$$

$$L_i(x) = \frac{\prod_{j\neq i\, j=0}^{n-1}(x - x_j)}{\prod_{j=0\, j\neq i}^{n-1}(x_i - x_j)} = \prod_{j=1\, j\neq i}^{n-1} \frac{x - x_j}{x_i - x_j}$$

$$P(X) = \sum_{i=0}^{n-1} y_i L_i(X)$$