CHAPTER 1

1.1 Modèles Discrets

Les modèles discrets considèrent que les changements de population se produisent à intervalles de temps distincts.

1.1.1 Equation Générale des Modèles Discrets

Considérons $N(t)$ comme la population d'individus à l'instant t. L'équation générale d'un modèle discret est donnée par la variation de la population entre deux instants discrets t et $t + \Delta t$:

$N(t+\Delta t) - N(t)$ = nombre de naissances − nombre de décès + nombre d'immigrations − nombre d'émigrations

En termes de taux, nous pouvons écrire:

$$N(t + \Delta t) - N(t) = n - m + i - e$$

où:

- n représente le nombre de naissances pendant l'intervalle Δt.
- m représente le nombre de décès pendant l'intervalle Δt.
- i représente le nombre d'immigrations pendant l'intervalle Δt.
- e représente le nombre d'émigrations pendant l'intervalle Δt.

1.1.2 Modèle de Croissance Géométrique

Le modèle de croissance géométrique est un modèle discret simple qui décrit la croissance d'une population dans des conditions idéales, où les ressources sont illimitées.

Hypothèses

- **Solde migratoire nul**: On suppose que le nombre d'immigrations est égal au nombre d'émigrations, donc $i - e = 0$.
- **Croissance proportionnelle à la taille de la population**: Le nombre de naissances est proportionnel à la taille de la population, avec un taux de natalité λ, et le nombre de décès est proportionnel à la taille de la population, avec un taux de mortalité μ. Ainsi, pendant l'intervalle Δt:
  - Nombre de naissances: $n = \lambda \Delta t N(t)$
  - Nombre de décès: $m = \mu \Delta t N(t)$

Équation et Solution

En posant $N_n = N(t_n)$ où $t_n = n\Delta t$, l'équation du modèle devient:

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

Soit en posant $z = \lambda - \mu$, le taux de croissance:

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

$$N_{n+1} = N_n + z\Delta t N_n = (1 + z\Delta t)N_n$$

En définissant le taux de croissance par unité de temps $c = z\Delta t$, on a:

$$N_{n+1} = (1 + c)N_n$$

La solution de cette équation de récurrence, avec condition initiale N_0 (taille initiale de la population), est:

$$N_n = (1 + c)^n N_0 = (1 + z\Delta t)^n N_0$$

Visualisation

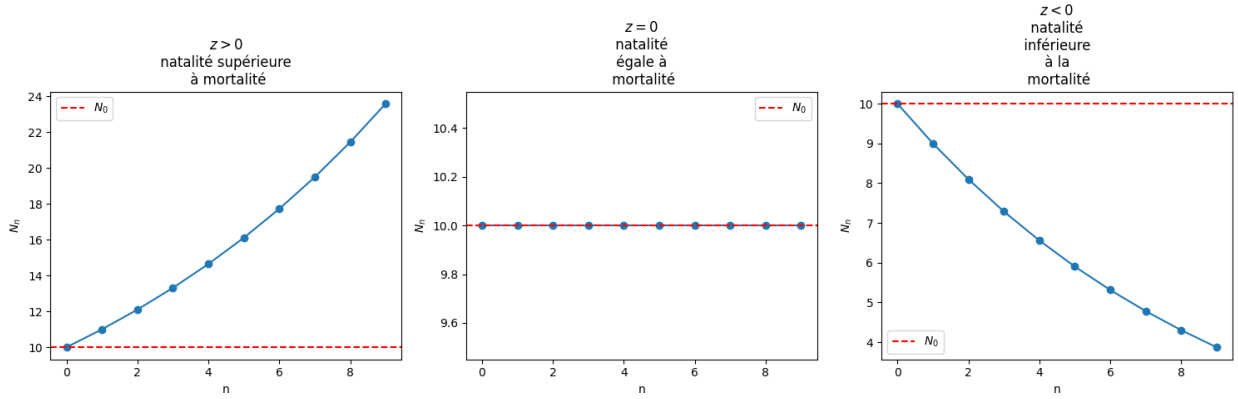

Figure 1.1: Visualisation du modèle de croissance géométrique pour différents taux de croissance z.

Propriétés du Modèle Géométrique

- **Tendance à l'infini pour** $z > 0$: Lorsque $z = \lambda - \mu > 0$, la population croît exponentiellement et tend vers l'infini lorsque $n \to \infty$. La solution $N(t) = N_0 e^{zt}$ est une approximation continue pour de petits Δt.
- **Croissance indéfinie pour** $z > 0$: Si $z > 0$, la population croît indéfiniment.
- **Extinction pour** $z < 0$: Si $z < 0$, la population décroît exponentiellement et tend vers l'extinction.
- **Population constante pour** $z = 0$: Si $z = 0$, la population reste constante au fil du temps, $N_n = N_0$ pour tout n.

Inconvénients du Modèle Géométrique

- **Croissance infinie irréaliste**: Une croissance infinie n'est pas réaliste dans le monde réel car les ressources sont limitées.
- **Approximation de partie entière**: Pour être rigoureux, on devrait écrire $E(\lambda \Delta t N_n)$ et $E(\mu \Delta t N_n)$ pour tenir compte du fait que le nombre d'individus doit être un entier, où $E(x)$ désigne la partie entière de x.

1.1.3 Modèle de croissance logistique discret

Le modèle de croissance logistique discret sera traité dans un exercice ultérieur.

1.2 Modèles Continus

Les modèles continus considèrent que les changements de population se produisent de manière continue dans le temps.

1.2.1 Motivation pour les Modèles Continus

Remark 1.1. L'utilisation de modèles continus est motivée par le fait que l'observation des populations sur des intervalles de temps très courts (Δt proche de 0) fournit beaucoup plus d'informations sur la dynamique de la population.

1.2.2 Modèle de Malthus

Le modèle de Malthus est le modèle continu le plus simple de croissance de population. Il est obtenu en passant à la limite du modèle géométrique lorsque $\Delta t \to 0$.

Hypothèses

- **Solde migratoire nul**: Comme pour le modèle géométrique, on suppose un solde migratoire nul.
- **Vitesses de natalité et de mortalité proportionnelles à la population**: On suppose que la vitesse de natalité et la vitesse de mortalité sont proportionnelles à la taille de la population à l'instant t.
  - Vitesse de natalité: $n(t) = \lambda N(t)$
  - Vitesse de mortalité: $m(t) = \mu N(t)$

Équation et Solution

Proposition 1.2. En reprenant l'équation de variation et en considérant les vitesses instantanées, l'équation différentielle du modèle de Malthus est:

$$N'(t) = \lim_{\Delta t \to 0} \frac{N(t+\Delta t) - N(t)}{\Delta t} = n(t) - m(t) = \lambda N(t) - \mu N(t)$$

Soit:

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

En posant $z = \lambda - \mu$, on obtient:

$$N'(t) = zN(t)$$

Avec la condition initiale $N(0) = N_0$, la solution de cette équation différentielle est:

$$N(t) = N_0 e^{zt} = N_0 e^{(\lambda - \mu)t}$$

Propriétés du Modèle de Malthus

- Similaire au modèle géométrique en termes de comportement qualitatif, mais décrit la croissance de manière continue.
- **Croissance exponentielle pour** $z > 0$: Si $z = \lambda - \mu > 0$, la population croît exponentiellement.
- **Population constante pour** $z = 0$: Si $z = \lambda - \mu = 0$, la population reste constante.

- **Décroissance exponentielle pour** $z < 0$: Si $z = \lambda - \mu < 0$, la population décroît exponentiellement vers zéro.

Inconvénients du Modèle de Malthus

- **Croissance exponentielle irréaliste**: Comme le modèle géométrique, le modèle de Malthus prédit une croissance exponentielle infinie, ce qui n'est pas réaliste à long terme en raison de la limitation des ressources.
- **Ne prend pas en compte la limitation des ressources et l'interaction avec l'environnement**.

1.2.3 Modèle de Verhulst (ou Logistique)

Le modèle de Verhulst est une amélioration du modèle de Malthus qui prend en compte la limitation des ressources.

Idée Centrale

Definition 1.3. Limiter la croissance de la population à un seuil maximal K, appelé *capacité biotique* ou *capacité de charge* du milieu.

Hypothèses

- **Solde migratoire nul**.
- **Taux de natalité fonction affine décroissante de la population**: Le taux de natalité diminue à mesure que la population approche de la capacité biotique K. On le modélise par une fonction affine décroissante: $\lambda = \lambda_0(1 - \frac{N(t)}{K})$, où λ_0 est le taux de natalité maximal (lorsque $N(t)$ est très petit).
- **Taux de mortalité fonction affine croissante de la population**: Le taux de mortalité augmente à mesure que la population approche de K. On le modélise par une fonction affine croissante: $\mu = \mu_0(1 + \frac{N(t)}{K})$, où μ_0 est le taux de mortalité minimal (lorsque $N(t)$ est très petit). Pour simplifier, on prend souvent μ constant. Dans les notes, il est considéré comme une fonction affine croissante $\mu = -\mu_1(1-\frac{N(t)}{K})$, ce qui implique que μ diminue quand $N(t)$ augmente, ce qui n'est pas biologiquement réaliste. On corrigera par $\mu = \mu_0 + \mu_1\frac{N(t)}{K} = \mu_0(1 + \frac{N(t)}{K})$ ou simplement μ constant.

En utilisant la version simplifiée avec μ constant et en posant $\lambda = \lambda_0(1 - \frac{N(t)}{K})$, et $z_0 = \lambda_0 - \mu$, le taux de croissance intrinsèque maximal, on obtient:

$$z = \lambda - \mu = \lambda_0(1 - \frac{N(t)}{K}) - \mu = (\lambda_0 - \mu) - \frac{\lambda_0}{K}N(t) = z_0 - \frac{\lambda_0}{K}N(t)$$

On approche souvent $\lambda_0 \approx z_0$, et on pose simplement $z \approx z_0(1 - \frac{N(t)}{K})$.

Équation et Solution

Proposition 1.4. L'équation différentielle du modèle de Verhulst est alors:

$$N'(t) = zN(t) = z_0\left(1 - \frac{N(t)}{K}\right)N(t)$$

Avec condition initiale $N(0) = N_0$. La solution de cette équation différentielle est donnée par:

$$N(t) = \frac{K}{1 + \left(\frac{K}{N_0} - 1\right) e^{-z_0 t}}$$

Visualisation

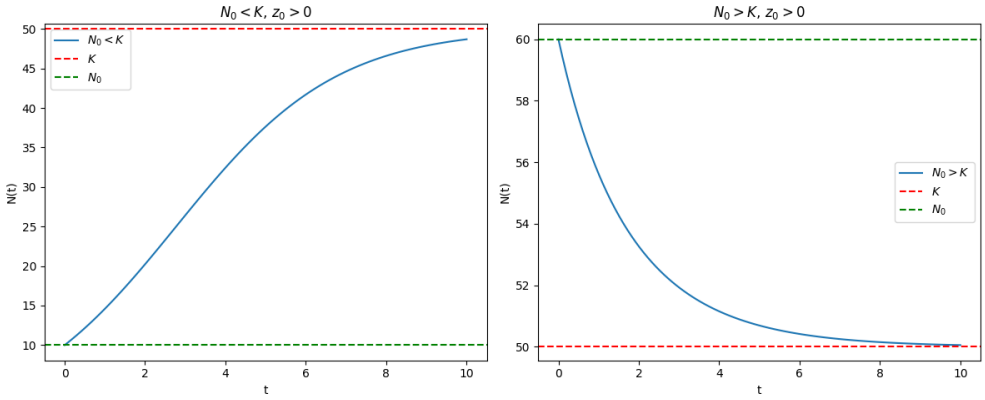

Figure 1.2: Visualisation du modèle de Verhulst pour différentes conditions initiales N_0 par rapport à la capacité biotique K.

1.3 Conclusion

Nous avons exploré les modèles discrets (géométrique) et continus (Malthus et Verhulst) pour la modélisation de populations. Le modèle géométrique et le modèle de Malthus, bien que simples, présentent des limitations importantes, notamment la prédiction d'une croissance infinie. Le modèle de Verhulst améliore ces modèles en introduisant la notion de capacité biotique, offrant une description plus réaliste de la dynamique des populations en tenant compte de la limitation des ressources.

CHAPTER 2

2.1 Introduction: Notion de Champ de Vecteurs et EDO

2.1.1 Généralités et Définitions

Nous allons étudier la notion de champ de vecteurs associé à une équation différentielle ordinaire (EDO).

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

Considérons une EDO du type :

$$y'(t) = f(t, y(t)), \quad t \in]t_0, T[, \tag{2.1}$$

avec la condition initiale :

$$y(t_0) = y_0, \tag{2.2}$$

où $y :]t_0, T[\to \mathbb{R}$ et $f :]t_0, T[\times \mathbb{R} \to \mathbb{R}$ est une fonction donnée, et $(t, x) \mapsto f(t, x)$.

2.2 Analyse Qualitative des Solutions d'EDO

Si l'on veut résoudre analytiquement une EDO, c'est-à-dire donner l'expression de $t \mapsto y(t)$, alors c'est terminé. Dans de nombreux cas, il suffit d'étudier la fonction $t \mapsto y(t)$.

Si l'on ne sait pas déterminer la solution analytique, on peut suivre une approche en deux étapes pour comprendre les solutions :

1. S'assurer de l'existence et de 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 par une simple analyse de $f(t, x)$. C'est ici qu'interviennent les champs de vecteurs.

2.3 Champ de Vecteurs: Définitions et Propriétés

2.3.1 Vecteur Tangent à une Courbe Paramétrée

Considérons une courbe paramétrée $t \mapsto \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ g(t) \end{pmatrix}$. Le vecteur tangent $\vec{v}$ à cette courbe est donné par :

$$\vec{v} = \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} = \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dx}\frac{dx}{dt} \end{pmatrix} = \frac{dx}{dt}\begin{pmatrix} 1 \\ \frac{dy}{dx} \end{pmatrix}$$
$$= \frac{1}{\frac{dt}{dx}}\begin{pmatrix} 1 \\ g'(x) \end{pmatrix} = \frac{1}{\dot{x}(t)}\begin{pmatrix} 1 \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{\dot{y}(t)}{\dot{x}(t)} \end{pmatrix} = \begin{pmatrix} 1 \\ \dot{y}(t) \end{pmatrix}$$

Si $x(t) = t$, alors $\dot{x}(t) = 1$, et le vecteur tangent devient $\vec{v} = \begin{pmatrix} 1 \\ \dot{y}(t) \end{pmatrix}$.

2.3.2 Lien entre Solution d'EDO et Vecteurs Vitesse

Proposition 2.1. y est solution de l'EDO $y'(t) = f(t, y(t))$ si et seulement si les vecteurs vitesses de la courbe paramétrée $t \mapsto \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} t \\ y(t) \end{pmatrix}$ au point t sont donnés par $u(t) = \begin{pmatrix} 1 \\ f(t, y(t)) \end{pmatrix}$.

2.3.3 Définition d'un Champ de Vecteurs

Definition 2.2. Soit $V : \mathbb{R}^2 \to \mathbb{R}^2$ un champ de vecteurs, défini par $(t, y) \mapsto V(t, y)$. Si ce champ de vecteurs est associé à l'EDO $y'(t) = f(t, y(t))$, alors $V(t, y) = \begin{pmatrix} 1 \\ f(t, y) \end{pmatrix}$.

2.4 Visualisation des Champs de Vecteurs (Implémentation Python)

2.4.1 Principe

Pour dessiner un champ de vecteurs, à chaque point $P = (P_x, P_y)$, on trace le vecteur $V \in V(P) = (V_x, V_y)$. On choisit une constante positive pour représenter les vecteurs trop longs.

Figure 2.1: Schéma de principe pour le dessin d'un champ de vecteurs.

2.4.2 Utilisation de `quiver` de Python

Pour implémenter le dessin de champs de vecteurs en Python, on utilise la fonction `quiver` de `matplotlib.pyplot`. Les arguments principaux sont : `plt.quiver(Px, Py, Vx, Vy, angles='xy', scale)`.

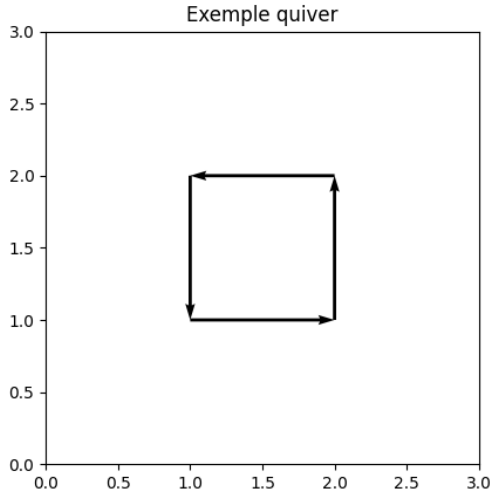

Figure 2.2: Exemple d'utilisation de quiver

2.4.3 Contrôle de la Longueur des Vecteurs

On peut ajouter un paramètre pour contrôler la longueur des vecteurs. Il est souvent nécessaire de normaliser les vecteurs pour une visualisation claire du champ de vecteurs.

Pour normaliser les vecteurs (V_x, V_y), on calcule d'abord la norme :

$$\text{norm} = \sqrt{V_x^2 + V_y^2} \tag{2.3}$$

Puis on normalise chaque composante :

$$V_x = V_x/\text{norm}$$
$$V_y = V_y/\text{norm}$$

2.5 Application: Recherche Approchée de Solutions (Python)

2.5.1 Objectif

On cherche une solution approchée de l'EDO $y'(t) = f(t, y(t))$, pour $t \in [t_0, t_0 + T]$ avec $y(t_0) = y_0$, en utilisant Python. Pour cela, il suffit de dessiner en quelques points où aboutit cette solution.

2.5.2 Méthode Python

Définissons un exemple de champ de vecteur avec $f(t, y) = r \cdot y \cdot (1 - y/k)$.

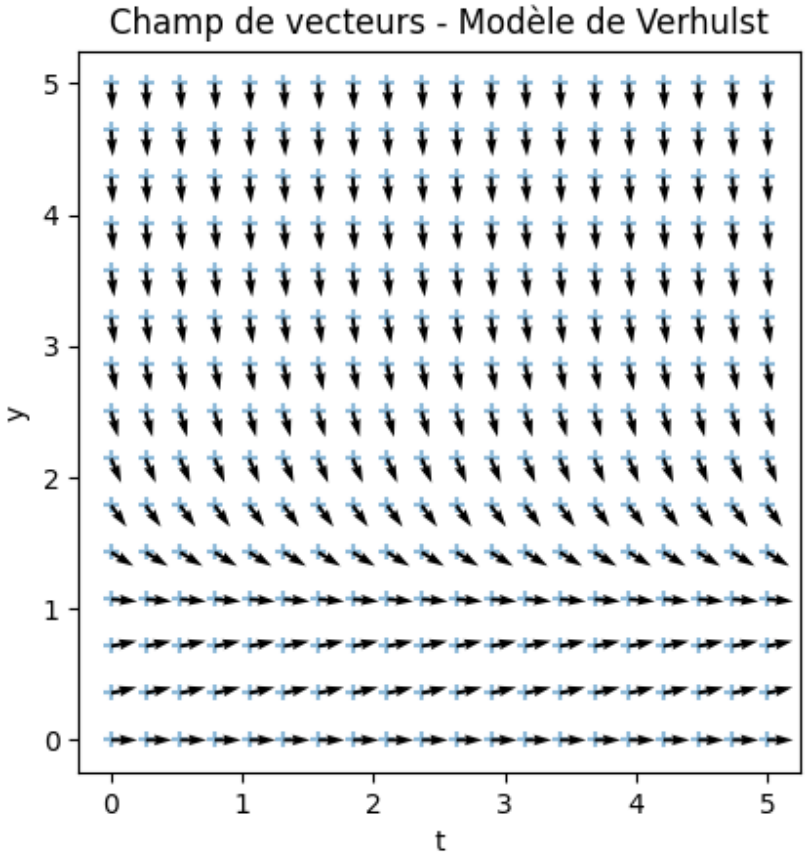

Figure 2.3: Champ de vecteurs pour le modèle de Verhulst.

2.6 Exploitation des Champs de Vecteurs pour Comprendre l'Allure des Solutions

Si l'on connaît les valeurs minimales et maximales de la solution, on peut avoir une idée de l'allure de la solution en observant le champ de vecteurs. Par exemple, si les vecteurs pointent vers le haut dans une certaine région, et vers le bas dans une autre, on peut déduire le comportement qualitatif des solutions dans ces régions.

CHAPTER 3

3.1 Analyse de la convergence

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

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

- avoir un polynôme construit et d'autant plus proche de la fonction que le nombre de points est grand.

C'est-à-dire, est-ce que la suite des "meilleurs" polynômes tend vers la fonction lorsque le nombre de points tend vers l'infini?

Question : Comment quantifier cette convergence? C'est-à-dire à quelle vitesse (ordre) cette convergence a lieu.

3.1.1 Approches

- **Approche 1 :** Approximation linéaire
  - Moindre carré de degré 1
- **Approche 2 :** Polynôme d'ordre 1
  - Interpolation polynomiale (Lagrange)
- **Approche 3 :** Autres approches
  - Splines, ondelettes, etc.

3.1.2 Valeur approchée par itération

Définition de convergence

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

Vitesse (ordre) de convergence

- **Convergence linéaire (ordre 1)**: Si $K_1 = \lim\limits_{n\to+\infty} \dfrac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|}$ existe et $K_1 \in [0, 1[$, on dit que la suite converge **linéairement** vers x^*, ou que la convergence est d'ordre 1.

- **Convergence quadratique (ordre 2)**: Si $K_1 = 0$, $K_2 = \lim_{n\to+\infty} \dfrac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^2}$ existe et non nul, on dit que la suite converge **quadratiquement** vers x^*, ou que la convergence est d'ordre 2.
- **Convergence d'ordre q**: Si $K_q = \lim_{n\to+\infty} \dfrac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q}$ existe et $\neq 0$, la convergence est d'ordre q.

Remark 3.2. La constante K_1 est appelée constante asymptotique d'erreur pour la convergence linéaire.

Exemples

Example 3.3. Soit $x_n = (0.2)^n$. On a $\lim_{n\to+\infty} x_n = 0$. La convergence est vers $x^* = 0$.

$$\lim_{n\to+\infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|} = \lim_{n\to+\infty} \frac{(0.2)^{n+1}}{(0.2)^n} = 0.2 \in [0, 1[$$

D'où, x_n converge à l'ordre 1. Sa constante asymptotique est $K_1 = 0.2$.

Example 3.4. Soit $y_n = (0.2)^{2^n}$.

$$y_{n+1} = (0.2)^{2^{n+1}} = (0.2)^{2^n \cdot 2} = ((0.2)^{2^n})^2 = (y_n)^2$$

$$\lim_{n\to+\infty} \frac{\|y_{n+1} - x^*\|}{\|y_n - x^*\|^2} = \lim_{n\to+\infty} \frac{y_{n+1}}{(y_n)^2} = \lim_{n\to+\infty} \frac{(y_n)^2}{(y_n)^2} = 1$$

D'où, convergence d'ordre 2, de constante $K_2 = 1$.

Définition formelle de la convergence d'ordre q

Definition 3.5. On dit que x_n converge vers x^* à l'ordre q s'il existe un entier $N \in \mathbb{N}$ et des constantes $A, B \in \mathbb{R}$ telles que pour tout $n > N$,

$$0 < A \leq \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} \leq B < +\infty$$

Remark 3.6. La convergence est au moins d'ordre q si seulement on a

$$\limsup_{n\to+\infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} < +\infty$$

3.1.3 Interprétation pratique de la vitesse de convergence

Nombre de chiffres significatifs

Remark 3.7. Si $|x_n - x^*| = 4 \cdot 10^{-8} = 0.\underbrace{00000004}_{8 \text{ digits}}$, on dit que x_n et x^* ont 8 chiffres exacts après la virgule.

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

On pose $d_n = -\log_{10}|x_n - x^*|$. d_n mesure approximativement le nombre de chiffres décimaux exacts entre x_n et x^*.

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

$$\begin{aligned}\log_{10}\|x_{n+1} - x^*\| &\approx \log_{10}(K_q\|x_n - x^*\|^q) \\ &= \log_{10} K_q + q\log_{10}\|x_n - x^*\|\end{aligned}$$

$$\begin{aligned}-d_{n+1} &\approx \log_{10} K_q + q(-d_n) \\ d_{n+1} &\approx qd_n - \log_{10} K_q\end{aligned}$$

Si $q = 1$, $d_{n+1} \approx d_n - \log_{10} K_1$. À chaque itération, on gagne environ $-\log_{10} K_1$ chiffres significatifs.

Si $q > 1$, $d_{n+1} \approx qd_n$. Le nombre de chiffres significatifs est approximativement multiplié par q à chaque itération.

3.1.4 Nombre d'itérations pour gagner un chiffre en convergence linéaire

Proposition 3.8. Si x_n converge à l'ordre 1 vers x^* avec une constante asymptotique K_1. Alors, le nombre d'itérations nécessaires pour gagner un chiffre significatif est approximativement $-\dfrac{1}{\log_{10}(K_1)}$.

Preuve. Soit m le nombre d'itérations pour gagner 1 chiffre significatif. Pour une convergence linéaire, $d_{n+1} \approx d_n - \log_{10} K_1$. En partant de d_n, après m itérations, on aura:

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

On souhaite gagner 1 chiffre significatif, donc $d_{n+m} \approx d_n + 1$.

$$d_n + 1 = d_n - m\log_{10} K_1 \Leftrightarrow 1 = -m\log_{10} K_1 \Leftrightarrow m = -\frac{1}{\log_{10} K_1}$$

□

3.1.5 Linéarisation pour estimation graphique de q

$$\begin{aligned}\frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|^q} &\approx K_q \\ \log_{10}\|x_{n+1} - x^*\| &\approx \log_{10}(K_q\|x_n - x^*\|^q) \\ \log_{10}\|x_{n+1} - x^*\| &\approx \underbrace{q\log_{10}\|x_n - x^*\|}_{x} + \underbrace{\log_{10} K_q}_{b}\end{aligned}$$

C'est de la forme $y = qx + b$, où $y = \log_{10}\|x_{n+1} - x^*\|$, $x = \log_{10}\|x_n - x^*\|$, q est la pente et $b = \log_{10} K_q$ est l'ordonnée à l'origine.

3.1.6 Procédure pour déterminer q graphiquement

1. Calculer les erreurs $\|x_n - x^*\|$ et $\|x_{n+1} - x^*\|$ pour les itérations disponibles.
2. Calculer les logarithmes (base 10) de ces erreurs: $\log_{10}\|x_n - x^*\|$ et $\log_{10}\|x_{n+1} - x^*\|$.
3. Tracer le nuage de points $(\log_{10}\|x_n - x^*\|, \log_{10}\|x_{n+1} - x^*\|)$.
4. Estimer graphiquement ou par régression linéaire la pente q de la droite qui approxime au mieux ce nuage de points. Cette pente q est une estimation de l'ordre de convergence.

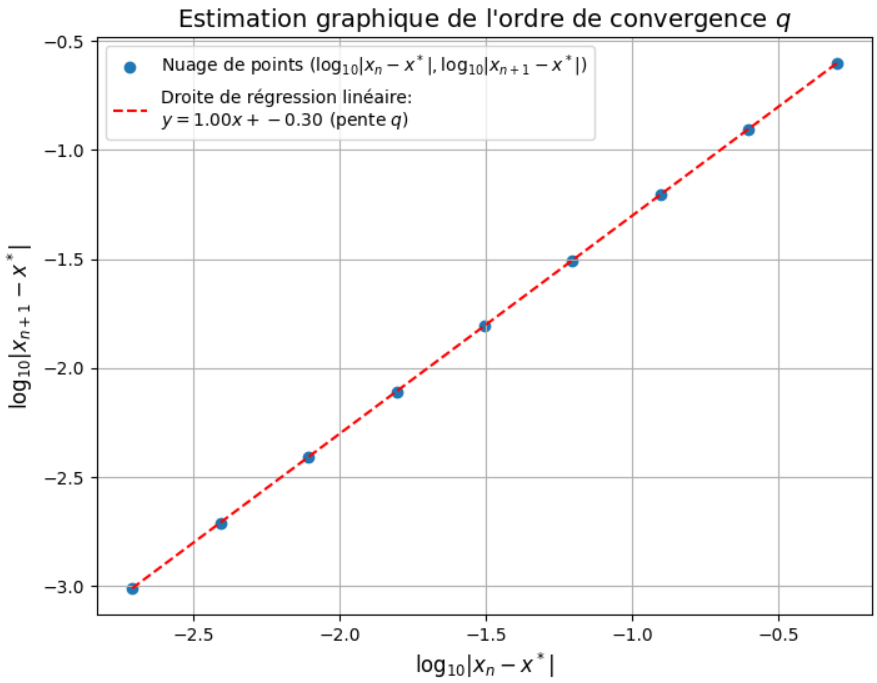

Figure 3.1: Estimation graphique de l'ordre de convergence q

3.2 Python code pour l'estimation de la convergence (exemple)

Listing 3.1: Code Python pour l'estimation de la convergence

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np

# Exemple de suite xn (remplacez avec votre suite)
xn = np.array([0.7**(i) for i in range(1, 20)])
x_star = 0 # Valeur limite de la suite

errors_n = np.abs(xn - x_star) # Erreur absolue |xn - x*|
log_errors_n = np.log10(errors_n) # Logarithme base 10 des erreurs

ex = log_errors_n[:-1] # $\log_{10} \|x_n - x^*\|$
ey = log_errors_n[1:]  # $\log_{10} \|x_{n+1} - x^*\|$

plt.figure(figsize=(8, 6))
plt.scatter(ex, ey, label="Nuage de points")

ab = np.polyfit(ex, ey, 1) # Regression lineaire (y = ax + b)
y_fit = ab[0]*ex + ab[1]
plt.plot(ex, y_fit, color='red', linestyle='--', label=f"Droite de r gression : $y = {ab[0


plt.xlabel("$\log_{10} \|x_n - x^*\|$", fontsize=12)
```

```
plt.ylabel("$\log_{10}\|x_{n+1} - x^*\|$", fontsize=12)
plt.title("Estimation graphique de l'ordre de convergence $q$", fontsize=14)
plt.legend()
plt.grid(True)
plt.show() # Affiche le graphique (pour execution hors LaTeX)
```

CHAPTER 4

4.1 Introduction à l'interpolation

4.1.1 Définition

Definition 4.1. Soit un nuage de points (exemple: un ensemble discret de points du graphe d'une fonction). Interpréter ce nuage de points correspond à chercher un polynôme de degré $N-1$ qui passe par chacun de ces points.

- Comment le construire ?
- $P_{N-1} \in \mathbb{P}_{N-1}[x]$
- $P_{N-1}(x_i) = y_i$

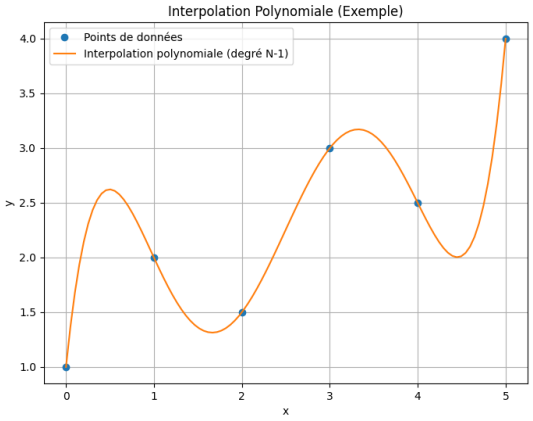

Figure 4.1: Illustration de l'interpolation polynomiale.

4.1.2 Motivations

- La solution d'un problème est fournie par une formule représentative : noyau de la chaleur (ex: convolution) et on cherche la solution en un nombre de points. $\Longrightarrow$ 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 travers ses valeurs en un nombre fini de points et on souhaite l'évaluer partout. $\Longrightarrow$ Interpolation.
- On peut utiliser l'interpolation dans :
  - la résolution numérique
  - la résolution numérique des Équations Différentielles Ordinaires (EDO)
  - la visualisation scientifique

Definition 4.2. Un tel polynôme est appelé **polynôme interpolateur de Lagrange** de degré $N-1$ de ces points.

4.1.3 Exemples d'interpolation

Théorème: Polynôme interpolateur de degré 1

Theorem 4.3. Soient (x_1, y_1) et (x_2, y_2) deux points distincts de $\mathbb{R}^2$. Il existe une unique droite D passant par ces deux points.

$$(x, y) \in D \iff (x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1) = 0$$

Si de plus, $x_1 \neq x_2$, il existe un unique polynôme de degré 1 (i.e., $P \in \mathbb{P}_1[x]$) tel que $y = P(x)$.
avec

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

pour des abscisses x_1, x_2 distinctes.

Example 4.4. Montrons que $M(x, y)$ est sur la droite (M_1M_2) si et seulement si les vecteurs $\overrightarrow{M_1M}$ et $\overrightarrow{M_1M_2}$ sont colinéaires. Soient $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ et $M(x, y)$.

$$M \in (M_1M_2) \Leftrightarrow \overrightarrow{M_1M} // \overrightarrow{M_1M_2}$$

$$\Leftrightarrow \det(\overrightarrow{M_1M}, \overrightarrow{M_1M_2}) = 0$$

$$\Leftrightarrow \begin{vmatrix} x - x_1 & x_2 - x_1 \\ y - y_1 & y_2 - y_1 \end{vmatrix} = 0$$

$$\Leftrightarrow (x - x_1)(y_2 - y_1) - (x_2 - x_1)(y - y_1) = 0$$

Si $x_1 \neq x_2$, alors on peut réécrire l'équation de la droite sous la forme $y = ax + b$:

$$\Longrightarrow y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$$

$$\Leftrightarrow y = P_1(x)$$

Remark 4.5. On a l'écriture équivalente de P_1 :

$$P_1(x) = \frac{x - x_2}{x_1 - x_2} y_1 + \frac{x - x_1}{x_2 - x_1} y_2 = \frac{y_1 - y_2}{x_1 - x_2} x + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} = a_1 x + a_0$$

c'est l'écriture dans la base $(1, x)$ de $\mathbb{P}_1[x]$ (base canonique).

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

$$= \underbrace{\frac{x-x_2}{x_1-x_2}}_{\ell_1(x)} y_1 + \underbrace{\frac{x-x_1}{x_2-x_1}}_{\ell_2(x)} y_2$$

C'est l'écriture dans la base (ℓ_1, ℓ_2) de $\mathbb{P}_1[x]$ (base de Lagrange).

Remark 4.6. $\ell_1(x_1) = 1, \ell_1(x_2) = 0$
$\ell_2(x_1) = 0, \ell_2(x_2) = 1$

- $P_1(x) = y_1 + \frac{y_2-y_1}{x_2-x_1}(x - x_1)$ C'est l'écriture dans la base $(1, x - x_1)$ de $\mathbb{P}_1[x]$ (base de Newton).

Exemple: méthode de calcul employée

Chercher le polynôme interpolateur de Lagrange aux points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

Méthode 1. $(x_1 \neq x_2, x_2 \neq x_3, x_1 \neq x_3)$
P_2 sera un polynôme de degré 2 :

$$P_2(x) = a_0 + a_1x + a_2x^2$$

Comme $P_2(x_i) = y_i$, pour $i = 1, 2, 3$, on a le système d'équations linéaires:

$$\begin{cases} P_2(x_1) = y_1 \\ P_2(x_2) = y_2 \\ P_2(x_3) = y_3 \end{cases} \implies \begin{cases} a_0 + a_1x_1 + a_2x_1^2 = y_1 \\ a_0 + a_1x_2 + a_2x_2^2 = y_2 \\ a_0 + a_1x_3 + a_2x_3^2 = y_3 \end{cases}$$

Matriciellement :

$$\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \implies \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix}^{-1}}_{H^{-1}} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

Matrice de Vandermonde mal-conditionnée mais facile à construire.

Remark 4.7. Pour 2 points :

$$H = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \implies H^{-1} = \frac{1}{x_2 - x_1} \begin{pmatrix} x_2 & -x_1 \\ -1 & 1 \end{pmatrix}$$

si $x_1 \neq x_2$.

Méthode 2. Base de Newton

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

$$\begin{cases} P_2(x_1) = y_1 \implies a_0 = y_1 \\ P_2(x_2) = y_2 \implies a_0 + a_1(x_2 - x_1) = y_2 \\ P_2(x_3) = y_3 \implies a_0 + a_1(x_3 - x_1) + a_2(x_3 - x_1)(x_3 - x_2) = y_3 \end{cases}$$

$$\implies \begin{cases} a_0 = y_1 \\ a_1 = \frac{y_2-y_1}{x_2-x_1} \\ a_2 = \frac{y_3-y_1-\frac{y_2-y_1}{x_2-x_1}(x_3-x_1)}{(x_3-x_1)(x_3-x_2)} = \frac{\frac{y_3-y_1}{x_3-x_1} - \frac{y_2-y_1}{x_2-x_1}}{(x_3-x_2)} = \frac{\frac{y_3-y_1}{x_3-x_1} - \frac{y_2-y_1}{x_2-x_1}}{x_3-x_2} \end{cases}$$

Cette construction est différentielle et facile à mettre à jour quand on rajoute un point supplémentaire (on rajoute uniquement une ligne).

On a donc :

$$a_0 = y_1, \quad a_1 = \frac{y_2 - y_1}{x_2 - x_1}, \quad a_2 = \frac{\frac{y_3 - y_1}{x_3 - x_1} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_2}$$

Le polynôme P_2 s'écrit donc :

$$P_2(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + \frac{\frac{y_3 - y_1}{x_3 - x_1} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_2}(x - x_1)(x - x_2)$$

	a_0	a_1	a_2
x_1	y_1		
x_2	y_1	$\frac{y_2 - y_1}{x_2 - x_1}$	
x_3	y_1	$\frac{y_2 - y_1}{x_2 - x_1}$	$\frac{\frac{y_3 - y_1}{x_3 - x_1} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_2}$

Table 4.1: Tableau des coefficients pour la base de Newton.

Construction facile et différentielle par différences divisées : ajout d'un terme.

Méthode 3. Base de Lagrange

$$P_2(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}y_1 + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}y_2 + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}y_3$$

$$P_2(x) = \sum_{i=1}^{3} \left(\prod_{\substack{j=1 \\ j \neq i}}^{3} \frac{x - x_j}{x_i - x_j} \right) y_i$$

$$= \ell_1(x)y_1 + \ell_2(x)y_2 + \ell_3(x)y_3$$

Remark 4.8. Pour deux points (x_1, y_1) et (x_2, y_2), le polynôme interpolateur de Lagrange de degré 1 est :

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

4.2 Polynôme interpolateur de Lagrange

4.2.1 Définitions et propriétés

Théorème: Existence et unicité

Theorem 4.9 (Existence et unicité)**.** Soient $x_1, \ldots, x_n$ des réels deux à deux distincts et $y_1, \ldots, y_n$ des réels quelconques. Il existe un unique polynôme $P \in \mathbb{P}_{n-1}[x]$ (i.e. de degré au plus $n - 1$) tel que $P(x_i) = y_i, \forall i = 1, \ldots, n$.

On dit que P est le **polynôme interpolateur de Lagrange** aux points $(x_1, y_1), \ldots, (x_n, y_n)$.

Preuve: Soit l'application linéaire $\Phi : \mathbb{P}_{n-1}[x] \to \mathbb{R}^n$ définie par

$$P \mapsto \begin{pmatrix} P(x_1) \\ \vdots \\ P(x_n) \end{pmatrix}$$

Montrons que Φ est injective. Si $\Phi(P) = 0$, alors $P(x_i) = 0$ pour tout $i = 1, \ldots, n$. Donc P a n racines distinctes $x_1, \ldots, x_n$. Comme P est un polynôme de degré au plus $n-1$ avec n racines, il s'ensuit que $P \equiv 0$. Donc Φ est injective.

Comme $\mathbb{P}_{n-1}[x]$ et $\mathbb{R}^n$ sont deux espaces vectoriels de même dimension n, une application linéaire injective est aussi bijective, donc un isomorphisme d'espaces vectoriels. La bijectivité de Φ assure l'existence et l'unicité du polynôme interpolateur.

Definition 4.10. Si f est une fonction continue sur $[a, b] \to \mathbb{R}$, et $x_1, \ldots, x_n \in [a, b]$ sont n points deux à deux distincts, alors l'unique polynôme $P \in \mathbb{P}_{n-1}[x]$ tel que $P(x_i) = f(x_i)$, pour $i = 1, \ldots, n$ est appelé **polynôme d'interpolation de Lagrange** de f aux points $x_1, \ldots, x_n$.

4.2.2 Estimation de l'erreur d'interpolation

Théorème: Erreur d'interpolation

Theorem 4.11 (Erreur d'interpolation). Soient $a < b$, $f : [a, b] \to \mathbb{R}$ une fonction continue, et $x_1, \ldots, x_n$ n points deux à deux distincts dans $[a, b]$. Soit P_n le polynôme d'interpolation de Lagrange de f aux points x_i. Si f est de classe $\mathcal{C}^n$ sur $[a, b]$, alors pour tout $x \in [a, b]$, il existe $\xi \in [a, b]$ tel que :

$$f(x) - P_n(x) = \frac{f^{(n)}(\xi)}{n!} \underbrace{\omega_n(x)}_{=\prod_{i=1}^{n}(x-x_i)}$$

où $\omega_n(x) = (x - x_1) \cdots (x - x_n)$.

Corollaire: Si $|f^{(n)}(x)|$ est bornée par M sur $[a, b]$ pour tout $x \in [a, b]$, alors $\forall x \in [a, b]$,

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

Preuve: (à faire)

4.2.3 Implémentation avec Python

```
from scipy.interpolate import lagrange

x = np.array([1, 2, 3]) #   remplacer par les valeurs de x_1, x_2, x_3
y = np.array([2, 3, 1]) #   remplacer par les valeurs de y_1, y_2, y_3
p = lagrange(x,y)
print(p) # affiche le polyn me
print(p(2.5)) #   value   le polyn me en x=2.5
```

4.3 Construction des polynômes d'interpolation de Lagrange

4.3.1 Interpolation dans la base canonique (Vandermonde)

Construction

Soit $P(x) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{P}_{n-1}[x]$. On cherche les coefficients $a_0, \ldots, a_{n-1}$ tels que $P(x_k) = y_k$ pour $k = 1, \ldots, n$.

$$\sum_{i=0}^{n-1} a_i x_k^i = y_k, \quad k = 1, \ldots, n$$

Ce qui conduit au système linéaire matriciel suivant :

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$V(x_1, \ldots, x_n)\mathbf{a} = \mathbf{y}$$

où $V(x_1, \ldots, x_n)$ est la matrice de Vandermonde.

C'est une matrice pleine, souvent mal conditionnée, mais facile à construire.

```
def VDM_Mat(x):
    n = len(x)
    V = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            V[i, j] = x[i]**j
    return V

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

4.3.2 Evaluation efficace : Algorithme de Horner

Proposition: Algorithme de Horner

Proposition 4.12. Soit $P(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ un polynôme. On définit la suite $(q_k)_{k=0}^n$ par :

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

Alors $q_n = P(x)$.

Exemple: $P(x) = x^2 + 2x + 1 = (x+1)^2$

Pour évaluer $P(2)$: $q_0 = a_0 = 1$ $q_1 = q_0 \times 2 + a_1 = 1 \times 2 + 2 = 4$ $q_2 = q_1 \times 2 + a_2 = 4 \times 2 + 1 = 9 = P(2) = 2^2 + 2 \times 2 + 1 = 9$

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

def IntVal_VDM (x, y, xx):
    a = VDM_Poly(x,y)
    YY = Horner(a[::-1], xx) # reverse a pour correspondre     l'ordre des coefficients dan
    return YY
```

4.3.3 Interpolation dans la base duale: Formule de Lagrange et points barycentriques

Construction

L'idée est de prendre pour base de $\mathbb{P}_{n-1}[x]$ l'image réciproque de la base canonique de $\mathbb{R}^n$ par l'application Φ définie dans le théorème d'existence et unicité. On cherche donc une base $\{\mathcal{L}_j\}_{j=1}^n$ de $\mathbb{P}_{n-1}[x]$ telle que

$$\mathcal{L}_j(x_i) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

On construit les polynômes de Lagrange $\mathcal{L}_j(x)$ comme suit :

$$\mathcal{L}_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} = \frac{\prod_{i \neq j}(x - x_i)}{\prod_{i \neq j}(x_j - x_i)}$$

Le polynôme interpolateur de Lagrange s'écrit alors :

$$P(x) = \sum_{j=1}^{n} y_j \mathcal{L}_j(x)$$

CHAPTER 5

5.1 Introduction à l'interpolation polynomiale

L'interpolation polynomiale est une technique fondamentale en analyse numérique qui consiste à trouver un polynôme qui passe par un ensemble donné de points. Plus précisément, étant donnés $n+1$ points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ où les x_i sont distincts, l'interpolation polynomiale cherche à construire un polynôme $P(x)$ de degré au plus n tel que $P(x_i) = y_i$ pour $i = 0, 1, \ldots, n$. Ce polynôme $P(x)$ est appelé le polynôme d'interpolation.

L'interpolation polynomiale a de nombreuses applications dans divers domaines tels que l'approximation de fonctions, l'intégration numérique, la résolution d'équations différentielles et le traitement de données expérimentales. Différentes méthodes existent pour construire ce polynôme d'interpolation, chacune ayant ses avantages et ses inconvénients en termes de complexité, de stabilité et de facilité d'implémentation. Nous allons explorer ici les méthodes de Lagrange et de Newton.

5.2 Interpolation de Lagrange

5.2.1 Formule de Lagrange

La formule d'interpolation de Lagrange est une manière explicite d'écrire le polynôme d'interpolation. Elle repose sur l'utilisation des polynômes de base de Lagrange.

Definition 5.1 (Polynômes de base de Lagrange)**.** Soient $x_0, x_1, \ldots, x_n$ des points distincts. Le polynôme nodal $\omega_{n+1}(x)$ est défini par :

$$\omega_{n+1}(x) = \prod_{i=0}^{n} (x - x_i)$$

et les polynômes de base de Lagrange $l_i(x)$ associés aux points $x_0, x_1, \ldots, x_n$ sont définis par :

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)}$$

pour $i = 0, 1, \ldots, n$.

On remarque que les polynômes de base de Lagrange vérifient la propriété suivante :

$$l_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Proposition 5.2 (Formule d'interpolation de Lagrange)**.** Le polynôme d'interpolation de Lagrange $P(x)$ de degré au plus n qui interpole les points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ est donné par :

$$P(x) = \sum_{i=0}^{n} y_i l_i(x)$$

Preuve. Pour vérifier que $P(x)$ est bien le polynôme d'interpolation, il suffit de montrer que $P(x_j) = y_j$ pour tout $j = 0, 1, \ldots, n$.

$$P(x_j) = \sum_{i=0}^{n} y_i l_i(x_j) = \sum_{i=0}^{n} y_i \delta_{ij} = y_j$$

De plus, chaque $l_i(x)$ est un polynôme de degré n, donc $P(x)$ est un polynôme de degré au plus n. □

5.2.2 Exemple

[Insérer un exemple si disponible dans les notes manuscrites, sinon, un exemple simple peut être construit ici.]

5.3 Erreur d'interpolation

5.3.1 Formule de l'erreur d'interpolation

L'erreur d'interpolation mesure la différence entre la fonction $f(x)$ que l'on cherche à interpoler et le polynôme d'interpolation $P(x)$.

Proposition 5.3 (Formule de l'erreur d'interpolation)**.** Soit $f \in C^{n+1}([a, b])$ et $P(x)$ le polynôme d'interpolation de Lagrange de degré au plus n interpolant f aux points $x_0, x_1, \ldots, x_n \in [a, b]$. Alors, pour tout $x \in [a, b]$, il existe un point $\xi_x \in [a, b]$ tel que :

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

5.3.2 Analyse de l'erreur

La formule de l'erreur montre que l'erreur d'interpolation dépend de deux facteurs principaux :

- La dérivée $(n+1)$-ième de la fonction f, $f^{(n+1)}(\xi_x)$. Si la dérivée $(n+1)$-ième de f est petite sur $[a, b]$, alors l'erreur d'interpolation sera petite.
- Le polynôme nodal $\omega_{n+1}(x) = \prod_{i=0}^{n} (x - x_i)$. La distribution des points d'interpolation $x_0, x_1, \ldots, x_n$ influence la magnitude de $\omega_{n+1}(x)$. Choisir des points d'interpolation de manière à minimiser $|\omega_{n+1}(x)|$ sur $[a, b]$ peut réduire l'erreur d'interpolation. Par exemple, les points de Chebyshev sont connus pour minimiser la norme infinie de $\omega_{n+1}(x)$ sur $[-1, 1]$.

5.3.3 Convergence

Pour assurer la convergence de l'interpolation polynomiale, c'est-à-dire que $P_n(x) \to f(x)$ lorsque $n \to \infty$, il ne suffit pas d'augmenter le degré du polynôme d'interpolation en utilisant des points équidistants. Le phénomène de Runge montre que pour certaines fonctions, l'interpolation polynomiale avec des points équidistants peut diverger entre les nœuds, même si la fonction est analytique. Cependant, si on choisit judicieusement les points d'interpolation, comme les points de Chebyshev, et si la fonction f est suffisamment régulière, on peut garantir la convergence de l'interpolation polynomiale.

5.4 Interpolation de Newton

5.4.1 Différences divisées

La méthode de Newton utilise les différences divisées pour construire le polynôme d'interpolation.

Definition 5.4 (Différences divisées). Les différences divisées d'ordre zéro sont définies par $f[x_i] = f(x_i)$. Les différences divisées d'ordre supérieur sont définies par la formule de récurrence :

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Proposition 5.5 (Formule d'interpolation de Newton). Le polynôme d'interpolation de Newton de degré au plus n s'écrit :

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, x_1, \ldots, x_n] \prod_{i=0}^{n-1} (x - x_i)$$

que l'on peut écrire sous la forme :

$$P(x) = \sum_{k=0}^{n} f[x_0, x_1, \ldots, x_k] \prod_{i=0}^{k-1} (x - x_i)$$

avec $\prod_{i=0}^{-1}(x - x_i) = 1$.

5.4.2 Algorithme de calcul des différences divisées

Les différences divisées peuvent être organisées dans un tableau triangulaire.

Listing 5.1: Calcul des différences divisées

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np

def DifferencesDivisees(x,y):
    n = len(x)
    d = np.zeros([n, n])
    for i in range(n):
        d[i, 0] = y[i]
    for j in range(1, n):
        for i in range(n - j):
            d[i, j] = (d[i+1, j-1] - d[i, j-1]) / (x[i+j] - x[i])
    return d
```

Remark 5.6. La fonction `DifferencesDivisees(x,y)` prend en entrée les abscisses `x` et les ordonnées `y` des points d'interpolation et retourne une matrice `d` contenant les différences divisées. La diagonale supérieure de cette matrice contient les coefficients $f[x_0], f[x_0, x_1], \ldots, f[x_0, x_1, \ldots, x_n]$ nécessaires pour la formule d'interpolation de Newton.

5.4.3 Évaluation du polynôme de Newton : Formule de Horner-Newton

Pour évaluer efficacement le polynôme de Newton, on utilise la formule de Horner-Newton.

Listing 5.2: Évaluation du polynôme de Newton (Horner-Newton)

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np

def HornerNewton(a,x,xx):
    n = len(a)
    yy = a[n-1]
    for i in range(n-2, -1, -1):
        yy = a[i] + (xx - x[i]) * yy
    return yy
```

Remark 5.7. La fonction `HornerNewton(a,x,xx)` prend en entrée le vecteur des coefficients des différences divisées `a` (diagonale de la matrice retournée par `DifferencesDivisees`), le vecteur des abscisses `x` des points d'interpolation, et un point `xx` où l'on souhaite évaluer le polynôme. Elle retourne la valeur du polynôme de Newton évalué en `xx`. Cette méthode est plus efficace pour évaluer le polynôme que l'évaluation directe de la formule de Newton.

5.5 Comparaison des méthodes et complexité

5.5.1 Complexité

- **Interpolation de Lagrange**: Le calcul des polynômes de base de Lagrange $l_i(x)$ et l'évaluation du polynôme d'interpolation de Lagrange nécessitent $\mathcal{O}(n^2)$ opérations pour un point donné x. Si l'on souhaite obtenir la forme développée du polynôme, la complexité est plus élevée.
- **Interpolation de Newton**: Le calcul des différences divisées nécessite $\mathcal{O}(n^2)$ opérations. L'évaluation du polynôme de Newton en utilisant la formule de Horner-Newton nécessite $\mathcal{O}(n)$ opérations par point. C'est une méthode efficace pour évaluer le polynôme une fois les différences divisées calculées.

5.5.2 Optimisation et ajout de points

- **Formule de Horner pour Newton**: La formule de Horner-Newton est cruciale pour l'efficacité de l'évaluation du polynôme de Newton. Elle réduit la complexité de l'évaluation à $\mathcal{O}(n)$ une fois les coefficients (différences divisées) sont connus.
- **Ajout d'un nouveau point**:
  - *Lagrange*: L'ajout d'un nouveau point d'interpolation nécessite de recalculer tous les polynômes de base de Lagrange et de refaire la somme. Cela peut être coûteux.
  - *Newton*: Si on ajoute un nouveau point (x_{n+1}, y_{n+1}), on peut facilement étendre le polynôme d'interpolation de Newton en calculant une différence divisée supplémentaire $f[x_0, x_1, \ldots, x_{n+1}]$ et en ajoutant un terme à la formule existante. Les différences divisées déjà calculées restent valides. C'est un avantage majeur de la méthode de Newton.

5.6 Conclusion

L'interpolation polynomiale est un outil puissant pour approximer des fonctions et traiter des données. Les méthodes de Lagrange et Newton offrent différentes approches pour construire et évaluer le polynôme d'interpolation. La méthode de Lagrange donne une formule explicite mais est moins pratique pour les calculs et l'ajout de nouveaux points. La méthode de Newton, basée sur les différences divisées, est efficace pour l'évaluation grâce à la formule de Horner-Newton et permet d'ajouter facilement de nouveaux points d'interpolation. Le choix de la méthode dépend du contexte et des besoins spécifiques de l'application.

CHAPTER 6

6.1 Polynômes de Tchebychev

6.1.1 Définition

On définit les polynômes de Tchebychev par récurrence :

- $T_0(x) = 1$
- $T_1(x) = x$
- $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1$

Les premiers polynômes de Tchebychev sont donc :

- $T_0(x) = 1$
- $T_1(x) = x$
- $T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1$
- $T_3(x) = 2xT_2(x) - T_1(x) = 2x(2x^2 - 1) - x = 4x^3 - 3x$

6.1.2 Expression trigonométrique

On a aussi l'expression trigonométrique suivante pour les polynômes de Tchebychev :

$$T_n(x) = \cos(n \arccos(x)), \quad x \in [-1, 1]$$

On vérifie pour $n = 0, 1, 2$:

- $T_0(x) = \cos(0 \arccos(x)) = \cos(0) = 1$
- $T_1(x) = \cos(1 \arccos(x)) = \cos(\arccos(x)) = x$
- $T_2(x) = \cos(2 \arccos(x)) = 2\cos^2(\arccos(x)) - 1 = 2x^2 - 1$

Pour vérifier la relation de récurrence, posons $\theta = \arccos(x)$, donc $x = \cos(\theta)$. Alors

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

On a utilisé la formule trigonométrique : $\cos(a)\cos(b) = \frac{1}{2}[\cos(a+b) + \cos(a-b)]$.

6.1.3 Propriétés

1. **Racines de $T_n(x)$**: $T_n(x) = 0 \Leftrightarrow \cos(n \arccos(x)) = 0$. Posons $x = \cos(\theta)$. Alors $\cos(n\theta) = 0 \Leftrightarrow n\theta = \frac{\pi}{2} + k\pi, k \in \mathbb{Z}$. Donc $\theta = \frac{\pi}{2n} + k\frac{\pi}{n}$. Pour avoir n racines distinctes dans $[-1, 1]$, on prend $k = 0, 1, \ldots, n-1$.
$$x_k = \cos\left(\frac{\pi}{2n} + k\frac{\pi}{n}\right), \quad k = 0, 1, \ldots, n-1$$
sont les n racines de $T_n(x)$ dans $[-1, 1]$.

2. $|T_n(x)| \leq 1$ pour $x \in [-1, 1]$. En effet, pour $x \in [-1, 1]$, $T_n(x) = \cos(n \arccos(x))$, et $|\cos(\cdot)| \leq 1$. De plus, $T_n(\cos(\frac{k\pi}{n})) = \cos(k\pi) = (-1)^k$. Donc $\max_{x\in[-1,1]} |T_n(x)| = 1$ atteint en $x'_k = \cos(\frac{k\pi}{n})$, $k = 0, \ldots, n$.

3. **Orthogonalité**: Les polynômes de Tchebychev sont orthogonaux pour le produit scalaire :
$$\langle f, g\rangle = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}} dx$$
En effet, posons $x = \cos(\theta)$, $dx = -\sin(\theta)d\theta$, $\sqrt{1-x^2} = \sin(\theta)$.
$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = \int_{\pi}^{0} \frac{\cos(n\theta)\cos(m\theta)}{\sin(\theta)}(-\sin(\theta))d\theta = \int_{0}^{\pi} \cos(n\theta)\cos(m\theta)d\theta$$
On sait que $\int_0^\pi \cos(n\theta)\cos(m\theta)d\theta = 0$ si $n \neq m$, et $\int_0^\pi \cos^2(n\theta)d\theta = \frac{\pi}{2}$ si $n \neq 0$, et $\int_0^\pi \cos^2(0)d\theta = \pi$. Donc
$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & \text{si } n \neq m \\ \pi & \text{si } n = m = 0 \\ \frac{\pi}{2} & \text{si } n = m \neq 0 \end{cases}$$

6.1.4 Application : Polynôme de meilleure approximation uniforme

Proposition 6.1. Soit P un polynôme de degré n avec coefficient dominant égal à 1, alors
$$\max_{x\in[-1,1]} |P(x)| \geq \max_{x\in[-1,1]} \left|\frac{1}{2^{n-1}} T_n(x)\right| = \frac{1}{2^{n-1}}$$
De plus, il y a égalité si et seulement si $P(x) = \frac{1}{2^{n-1}} T_n(x)$. On dit que $\frac{1}{2^{n-1}} T_n(x)$ est le polynôme de Tchebychev normalisé.

Soient $x_0, \ldots, x_n$ des points 2 à 2 distincts de $[-1, 1]$. On a :
$$\max_{x\in[-1,1]} \prod_{i=0}^{n} |x - x_i| \geq \max_{x\in[-1,1]} \prod_{i=1}^{n} |x - x_i^*|$$
avec x_i^* les racines de $T_{n+1}(x)$ translatées et dilatées sur $[-1, 1]$ (racines de Tchebychev).
$$x_k^* = \cos\left(\frac{\pi}{2(n+1)} + \frac{k\pi}{n+1}\right), \quad k = 0, \ldots, n$$
sont les racines de T_{n+1}.

6.1.5 Application à l'interpolation polynomiale

Soient $x_0, \ldots, x_n$ $n+1$ points 2 à 2 distincts, f une fonction $n+1$ fois continûment dérivable. Soit $P_n(x)$ le polynôme d'interpolation de Lagrange de f aux points x_i. Alors l'erreur d'interpolation est donnée par :
$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \quad \xi_x \in [\min(x, x_i), \max(x, x_i)]$$

Donc

$$|f(x) - P_n(x)| \leq \frac{\max_{\xi \in [a,b]} |f^{(n+1)}(\xi)|}{(n+1)!} \max_{x \in [a,b]} \prod_{i=0}^{n} |x - x_i|$$

Pour minimiser l'erreur d'interpolation, il faut minimiser $\max_{x \in [a,b]} \prod_{i=0}^{n} |x - x_i|$. D'après le corollaire précédent, les points de Tchebychev minimisent ce terme (à translation et dilatation près pour adapter l'intervalle $[-1, 1]$ à $[a, b]$).

6.2 Intégration numérique

6.2.1 Motivation et concept général

On cherche à approcher numériquement l'intégrale d'une fonction f sur un intervalle $[a, b]$:

$$I(f) = \int_a^b f(x)dx$$

On approche $I(f)$ par une somme pondérée de valeurs de f en certains points $x_i \in [a, b]$:

$$Q_n(f) = \sum_{i=0}^{n} \omega_i f(x_i)$$

où x_i sont les **nœuds** de quadrature et ω_i sont les **poids** de quadrature. On cherche à construire des formules de quadrature $Q_n(f)$ qui soient exactes pour les polynômes de degré le plus élevé possible.

Definition 6.2. On dit qu'une formule de quadrature $Q_n(f) = \sum_{i=0}^{n} \omega_i f(x_i)$ est de degré de précision r si elle est exacte pour tous les polynômes de degré $\leq r$, et n'est pas exacte pour au moins un polynôme de degré $r + 1$. C'est-à-dire :

- $\forall P \in \mathbb{P}_r, \quad Q_n(P) = I(P) = \int_a^b P(x)dx$
- $\exists P \in \mathbb{P}_{r+1}, \quad Q_n(P) \neq I(P) = \int_a^b P(x)dx$

6.2.2 Construction des formules de quadrature

Idée : utiliser l'interpolation polynomiale. Soient $x_0, \ldots, x_n$ $n + 1$ points distincts dans $[a, b]$. Soit $P_n(x)$ le polynôme d'interpolation de Lagrange de f aux points x_i. On approche $I(f)$ par $I(P_n) = \int_a^b P_n(x)dx$. On sait que $P_n(x) = \sum_{i=0}^{n} f(x_i)L_i(x)$ où $L_i(x)$ sont les polynômes de Lagrange :

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

Donc

$$Q_n(f) = I(P_n) = \int_a^b P_n(x)dx = \int_a^b \sum_{i=0}^{n} f(x_i)L_i(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_i(x)dx$$

On pose $\omega_i = \int_a^b L_i(x)dx$. Alors $Q_n(f) = \sum_{i=0}^{n} \omega_i f(x_i)$ est une formule de quadrature.

Proposition 6.3. La formule de quadrature $Q_n(f)$ construite à partir de l'interpolation de Lagrange aux points $x_0, \ldots, x_n$ est de degré de précision au moins n.

Preuve. Si $P \in \mathbb{P}_n$, alors $P_n(x) = P(x)$ (le polynôme d'interpolation d'un polynôme de degré $\leq n$ est lui-même). Donc $Q_n(P) = \int_a^b P_n(x)dx = \int_a^b P(x)dx = I(P)$. Donc Q_n est exacte pour les polynômes de degré $\leq n$. □

6.2.3 Exemples

Example 6.4 (Formule du point milieu). $n = 0$, un seul point $x_0 = \frac{a+b}{2}$ (milieu de l'intervalle). $L_0(x) = 1$. $\omega_0 = \int_a^b L_0(x)dx = \int_a^b 1dx = b - a$. $Q_0(f) = (b-a)f\left(\frac{a+b}{2}\right)$. Degré de précision : 1. Exacte pour les polynômes de degré ≤ 1. Exemple : $\int_0^1 xdx = \frac{1}{2}$. $Q_0(x) = (1-0) \times \frac{0+1}{2} = \frac{1}{2}$. Exact. $\int_0^1 x^2dx = \frac{1}{3}$. $Q_0(x^2) = (1-0) \times \left(\frac{0+1}{2}\right)^2 = \frac{1}{4} \neq \frac{1}{3}$. Non exacte pour degré 2.

Example 6.5 (Formule des trapèzes). $n = 1$, deux points $x_0 = a$, $x_1 = b$. $L_0(x) = \frac{x-x_1}{x_0-x_1} = \frac{x-b}{a-b}$, $L_1(x) = \frac{x-x_0}{x_1-x_0} = \frac{x-a}{b-a}$. $\omega_0 = \int_a^b \frac{x-b}{a-b}dx = \frac{1}{a-b}\left[\frac{x^2}{2} - bx\right]_a^b = \frac{1}{a-b}\left[\left(\frac{b^2}{2} - b^2\right) - \left(\frac{a^2}{2} - ba\right)\right] = \frac{1}{a-b}\left[-\frac{b^2}{2} - \frac{a^2}{2} + ba\right] = \frac{b-a}{2}$. $\omega_1 = \int_a^b \frac{x-a}{b-a}dx = \frac{1}{b-a}\left[\frac{x^2}{2} - ax\right]_a^b = \frac{1}{b-a}\left[\left(\frac{b^2}{2} - ab\right) - \left(\frac{a^2}{2} - a^2\right)\right] = \frac{1}{b-a}\left[\frac{b^2}{2} - ab + \frac{a^2}{2}\right] = \frac{b-a}{2}$. $Q_1(f) = \frac{b-a}{2}[f(a) + f(b)]$. Degré de précision : 1. Exacte pour les polynômes de degré ≤ 1. Exemple : $\int_0^1 x^2dx = \frac{1}{3}$. $Q_1(x^2) = \frac{1-0}{2}[0^2 + 1^2] = \frac{1}{2} \neq \frac{1}{3}$. Non exacte pour degré 2.

6.2.4 Estimation de l'erreur

Soit $Q_n(f) = \sum_{i=0}^n \omega_i f(x_i)$ la formule de quadrature construite par interpolation de Lagrange aux points $x_0, \ldots, x_n$. On sait que l'erreur d'interpolation est :

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^n (x - x_i)$$

Donc l'erreur de quadrature est :

$$E_n(f) = I(f) - Q_n(f) = \int_a^b f(x)dx - \int_a^b P_n(x)dx = \int_a^b [f(x) - P_n(x)]dx$$
$$= \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^n (x - x_i)dx = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b \prod_{i=0}^n (x - x_i)dx$$

Si on suppose que $f^{(n+1)}$ est continue, on peut utiliser la formule de la moyenne pour l'intégrale :

$$E_n(f) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b \prod_{i=0}^n (x - x_i)dx, \quad \xi \in [a, b]$$

où ξ est une valeur intermédiaire dans $[a, b]$. En pratique, on borne l'erreur :

$$|E_n(f)| \leq \frac{\max_{\xi \in [a,b]} |f^{(n+1)}(\xi)|}{(n+1)!} \int_a^b \prod_{i=0}^n |x - x_i|dx$$

Remark 6.6. Pour la formule du point milieu sur $[-1, 1]$, $x_0 = 0$, $n = 0$. $Q_0(f) = 2f(0)$. $\int_{-1}^1 (x - x_0)dx = \int_{-1}^1 xdx = 0$. Donc la formule d'erreur simple ne s'applique pas directement car $\int_a^b \prod_{i=0}^n (x - x_i)dx = 0$ dans certains cas (comme ici). Il faut une formule d'erreur plus précise.