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.

2 Interpolation de Lagrange

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 2.1 (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 2.2 (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. □

2.2 Exemple

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

3 Erreur d'interpolation

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 3.1 (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 :

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

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 $[-1,1]$.

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.

4 Interpolation de Newton

4.1 Différences divisées

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

Definition 4.1 (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 4.2 (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 $\prod_{i=0}^{-1}(x - x_i) = 1$.

4.2 Algorithme de calcul des différences divisées

Les différences divisées peuvent être organisées dans un tableau triangulaire.

Listing 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 4.3. 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.

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 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 4.4. 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 Comparaison des méthodes et complexité

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.

5.2 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.

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.