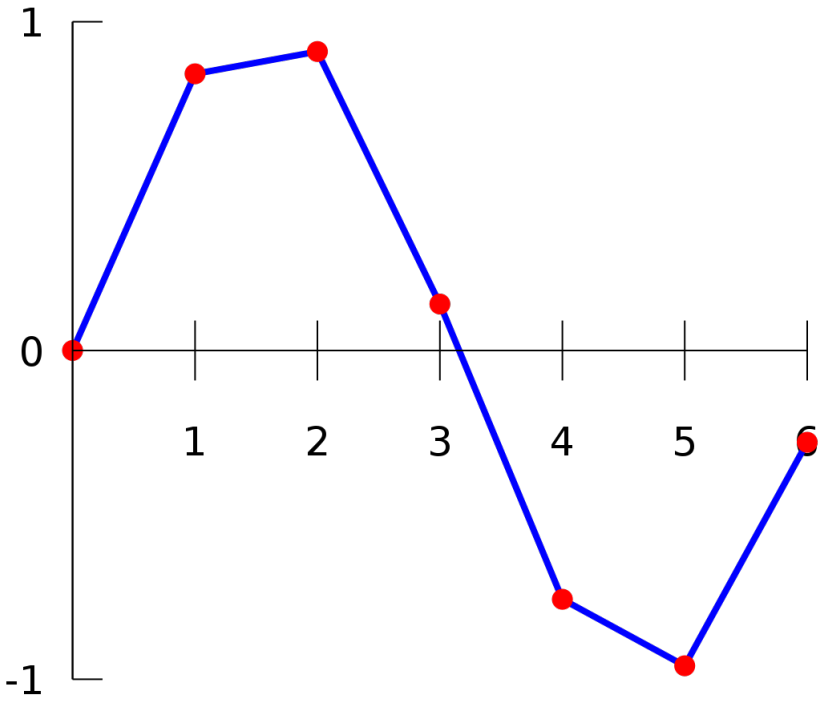
HEPIA

Interpolation

Math

Dos Reis Cédric & De Biasi Loris



17 décembre 2018

Table des matières

1 Introduction

Ce travail a pour but d'approfondir nos connaissances des méthodes d'interpolation vues en cours.

Ce travail se divise en 3 parties :

1. Interpolation 1d avec 3 méthodes
   - Polynomiale de degré N-1
   - Continue par morceaux de degré 1
   - Spline scellé
2. Calculer l'erreur
3. Interpolation 2d avec les 3 mêmes méthodes

2 Les 3 méthodes

Nous avons choisi la fonction suivante pour les interpolations :

$$f(x) = \frac{1}{1+x^2}$$

Pour chacune des méthodes nous avons besoin de calculer la différence divisée. Voici la formule qui permet de la calculer :

$$Ordre0 : [y0] = y0$$

$$Ordre1 : [y0, y1] = \frac{y1 - y0}{x1 - x0}$$

$$Ordre2 : [y0, y1, y2] = \frac{\frac{y2-y1}{x2-x1} - \frac{y1-y0}{x1-x0}}{x2 - x0}$$

2.1 Interpolation polynomiale de degré N-1

Cette méthode consiste à trouver un polynôme qui passe par un ensemble de points pour pouvoir en calculer d'autre. Pour ce faire, la formule de différence divisée ainsi que la formule de Newton sont utilisés.

$$\begin{aligned} p_n(x) = y[x_0] + (x - x_0)\delta y[x_0, x_1] + \dots \\ + (x - x_0)(x - x_1)\cdots(x - x_{n-1})\delta y^n[x_0, \dots, x_n]. \end{aligned} \tag{1}$$

2.2 Interpolation continue par morceaux de degré 1

Cette méthode consiste à relier les points par des droites pour ensuite pouvoir calculer de nouveaux points.

Dans un premier temps, on calcule la pente de la droite

$$a_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

On calcule ensuite l'ordonnée à l'origine

$$b_i = -\frac{x_i y_{i+1} - x_{i+1} y_i}{x_{i+1} - x_i}$$

Il ne reste plus qu'a injecter le position X que l'on souhaite.

$$f_h(x) = a_i x + b_i$$

2.3 Interpolation spline scellé

Cette méthode consiste à déterminer un polynome $s(x)$ de degré 3 définit sur $[x_i, x_{i+1}]$

$$\begin{aligned} s_i(x) = y_i + \delta y[x_{i+1}, x_i](x - x_i) + \frac{1}{2(x_{i+1} - x_i)} (p_{i+1} - p_i)(x - x_i)(x - x_{i+1}) \\ + \frac{1}{2(x_{i+1} - x_i)^2} (p_{i+1} + p_i - 2\delta y[x_{i+1}, x_i]) \left((x - x_i)^2(x - x_{i+1}) + (x - x_i)(x - x_{i+1})^2\right), \end{aligned} \tag{2}$$

Cependant $[p_i, p_{i+1}]$ ne sont pas connus, voici le système de $N-1$ équations sur $N-1$ inconnues qui permet de les obtenir :

$$p_0 = p_n = 0$$

$$p_{i-1} + 4p_i + p_{i+1} = 3(\delta y[x_{i+1}, x_i] + \delta y[x_i, x_{i-1}]), \quad i = 1, ..., N-1.$$

Le système d'équation est ensuite résolu avec la méthode *Gaussian Elimination*

2.4 Calcul de l'erreur

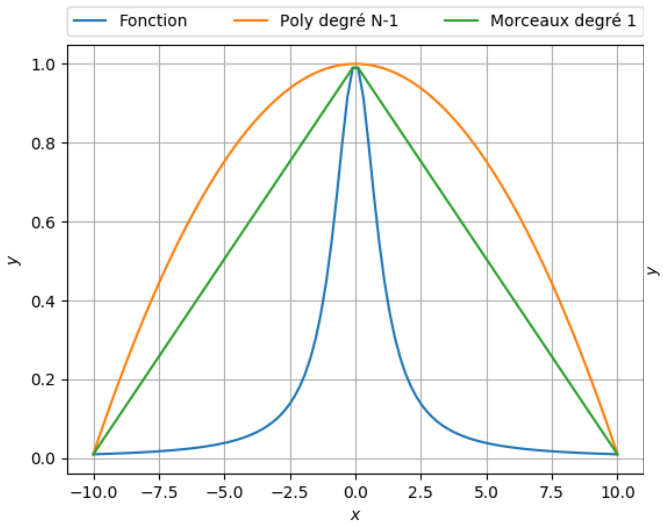
Pour calculer l'erreur d'une fonction, nous avons calculé plusieurs X donné et comparé les résultats de chaque méthode avec les valeurs obtenues en utilisant la fonction.

```
Erreur = 0
for Un intervalle donné do
 | Erreur += | résultatFonction[index] - résultatMéthode[index] |
end
retourne Erreur
```

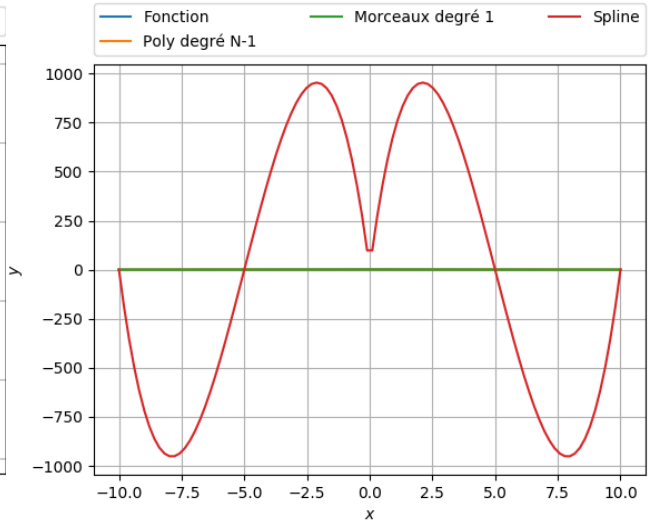
3 Résultat

3.1 Valeurs obtenues

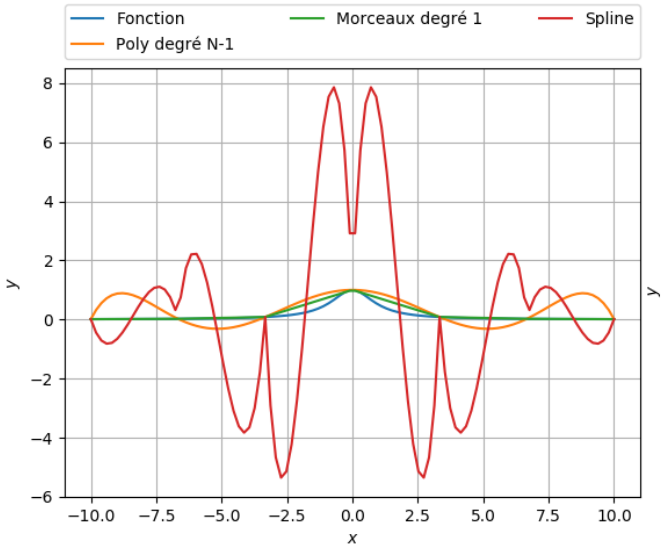
Voici les valeurs obtenues sous forme de graphique. En dessous de chaque image se trouve le nombre de points utilisé pour obtenir ce résultat :



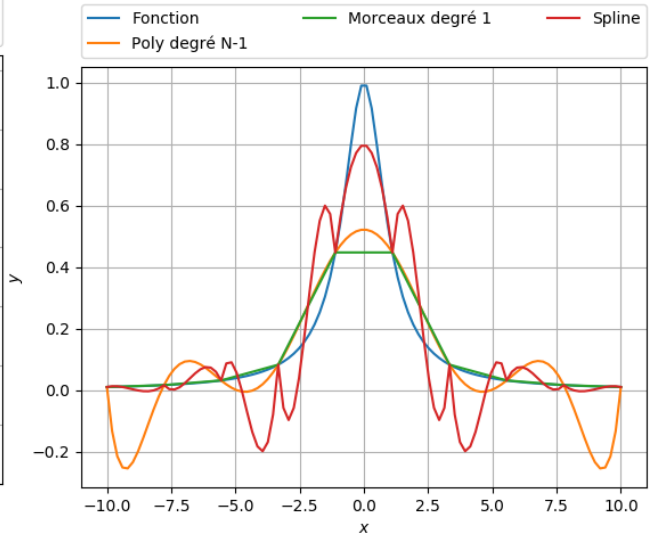
(a) 3 points sans spline



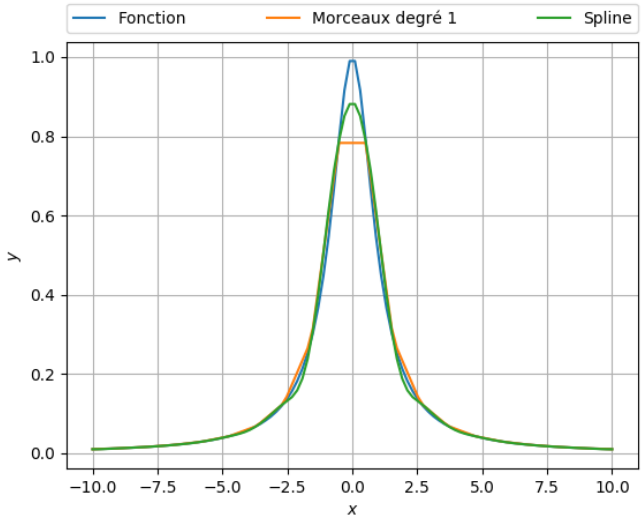
(b) 3 points avec spline



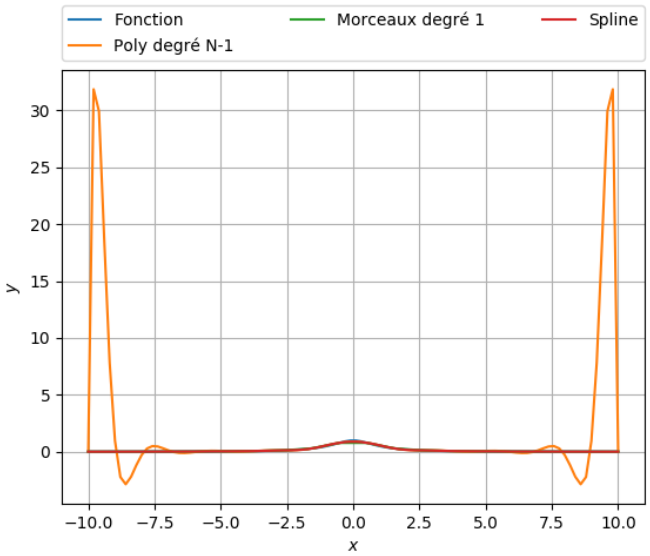
(c) 7 points



(d) 10 points



(e) 20 points sans polynomiale



(f) 20 points avec polynomiale

Dans l'image (a), la courbe de la méthode Spline a été cachée car elle empêchait de voir correctement les autres (cf. 2.2 Erreur), ainsi que la méthode polynomiale dans l'image (e).

Dans un premier temps, on se rend compte que la méthode polynomiale est beaucoup trop aléatoire. À partir d'un certain nombre de points dépassé, elle donne de très mauvais résultat.

Pour ce qui est de la méthode continue par morceaux de degré 1, on se rend compte qu'au plus il y a de point, moins il y a d'erreur, mais l'erreur diminue très lentement.

Quant à la méthode spline scellée, l'erreur est énorme quand il y a peu de points, mais elle s'améliore très vite.

3.2 Erreur

En utilisant la méthode citée précédemment, nous avons calculé l'erreur pour plusieurs nombres de points donnés différents. On constate la même chose que dans les graphiques obtenus. On se rend très clairement compte que la méthode polynomiale est très aléatoire, la méthode continue par morceaux est de plus en plus précise, mais prend pas mal de temps et finalement la méthode spline scellée qui donne un très mauvais résultat en ayant peu de points, mais s'améliore très vite.

Pour conclure, en dessous d'un certain nombre de points il veut mieux utiliser la méthode continue par morceaux, alors qu'en dessus il vaut mieux utiliser la méthode spline. La méthode polynomiale n'est (dans ce cas) jamais une bonne solution.

Nombre de point	**3**	**7**	**10**	**20**
Polynomiale	0.1167411775	4.112609259	1.725884407	99.736026078
Continue morceaux degré 1	0.0587555810	0.005850899	0.004040675	0.0032539311
Spline scellé	540.85851337	3.931177076	0.084408381	0.0020256244

3.3 Résultat interpolation 2d

Pour faire de l'interpolation en deux dimensions, il suffit d'utiliser l'interpolation à une dimension dans les deux directions, permettant ainsi de créer les nouvelles lignes ainsi que les nouvelles colonnes. Voilà grossièrement l'algorithme que nous avons utilisé pour effectuer l'interpolation en deux dimensions pour chaque méthode.

```
nouvelleImage = tableau vide
// Génère une nouvelle colonne
for Chaque ligne de l'image do
    ajouter le pixel courant à nouvelleImage
    for Chaque colonne de l'image do
        calculer le nouveau pixel en utilisant l'indice de ligne
        ajouter le nouveau pixel à nouvelleImage
    end
end
// Génère une nouvelle ligne
for Chaque ligne de nouvelleImage do
    for Chaque colonne de nouvelleImage do
        calculer le nouveau pixel en utilisant l'indice de colonne
        ajouter le nouveau pixel à nouvelleImage
    end
end
// Génère la dernière ligne
for Chaque colonne de nouvelleImage do
    calculer le nouveau pixel en utilisant l'indice de colonne
    ajouter le nouveau pixel à nouvelleImage
end
```

FIGURE 1 – Image de base

(a) Continue par morceaux (b) Polynomiale de degré N-1 (c) Spline scellée

Comme il est possible de le voir juste au-dessus, la méthode continue par morceaux ainsi que la méthode spline semble donner un résultat qui a l'air tout à fait correct, alors que la méthode polynomiale donne l'impression que l'on a appliqué un filtre à l'image.

4 Application

4.1 Interpolation 1D

L'utilisation de ces méthodes sur une dimension permet de déterminer une fonction qui passe par un nombre de point (x_i, y_i) définit dans un interval, cela donne ensuite la possibilité d'approximer d'autre points non définit dans l'interval.

4.2 Interpolation 2D

Nous utilisons l'interpolation sur 2 dimensions pour agrandir des images en calculant les nouvelles lignes et colonnes de pixels de l'image agrandi en fonction de l'image original. Pour ce faire, il faut utiliser l'interpolation 1D sur chaque colonne et chaque lignes.

5 Conclusion

Pour conclure, nous avons remarqué que selon le nombre de points donné et la forme de la fonction, il est préférable d'utiliser une méthode plutôt qu'une autre. Si dans le futur nous devions utiliser une méthode, nous privilégierons la méthode continue par morceaux ou la méthode spline qui nous ont donné les meilleurs résultats. Ce travail nous aura permis d'appliquer la théorie vue en cours et d'approfondir nos connaissances en interpolation.