Interpolation et approximation

Valentin PORTAIL et Jérémie VILLEPREUX

10 novembre 2022

Table des matières


Figure 1 – Joseph-Louis Lagrange et Eric Harold Neville

1 Pré-requis

1.1 Les bibliothèques nécessaires

Ce projet nécessite l'installation des trois bibliothèques standards suivantes :
- `stdio.h`, pour la gestion des affichages et saisie clavier ;
- `stdlib.h`, pour la gestion des allocations dynamiques ;
- `math.h`, pour l'utilisation de la fonction puissance ;

1.2 Comment compiler notre projet ?

Un fichier `Makefile` est également disponible pour compiler le programme. Le compilateur choisi est `gcc` et l'exécutable généré aura pour nom `prog`. Il s'obtient avec la commande :

```
1 $ make
```

Il est également possible de compiler le programme sans utiliser le `Makefile` à disposition avec la commande :

```
1 $ gcc -Wall -Wextra -o prog *.c -lm
```

Remarque 1. La compilation de tous les fichiers sources est nécessaire à l'obtention de l'exécutable.

Remarque 2. Il y a également un répertoire sous le nom de `Trace Python` qui contient du code `Python`. Pour lancer le code, il est nécessaire d'avoir les librairies `mathplotlib.pyplot` et `numpy`. Mais l'exécution n'est pas nécessaire (car déjà fait).

2 Rappel rapide des méthodes

Les méthodes que nous allons voir permettent de retrouver l'équation d'un polynôme qui traverse une liste de points de dimension 2 donnée.

Il y a deux principaux types de méthodes :

L'interpolation : On cherche la fonction qui passe exactement par tous les points initiaux.

L'approximation : On recherche non pas une fonction qui passe par tous les points, mais une qui représente au mieux l'ensemble des points, c'est-à-dire, une fonction dont l'écart entre les valeurs obtenues et celles désirées est minimal.

2.1 Interpolation

On recherche une fonction polynomiale de la forme $P_{(n-1)}(x) = a_{(n-1)}x^{(n-1)} + a_{(n-2)}x^{(n-2)} + \ldots + a_1x + a_0$ pour n points.

Le théorème de l'*unisolvance* nous permet de montrer que la fonction polynomiale obtenue est unique et vérifie les hypothèses de l'interpolation.

Nous allons implémenter deux méthodes d'interpolation : celle de Lagrange et celle de Neville.

2.1.1 Méthode de Lagrange

Nous allons implémenter le polynôme $P_{(N-1)}$ sous la forme :

$$P_{(N-1)}(x) = \sum_{i=0}^{(N-1)} y_i l_i(x)$$

où $l_i(x)$ est une fonction cardinale définie par :

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

La fonction $l_i(x)$ est un produit de $N-1$ polynômes de degré 1. C'est donc un polynôme de degré $N-1$.

L'objectif de la méthode de Lagrange est de construire le polynôme de degré $N-1$ d'un ensemble de N points par l'équation plus haut.

Le nombre d'opérations pour l'écriture avec la méthode de Lagrange est de $2N(N+1)$. Le coût de l'évaluation en une valeur est de $N(2N+3)$.

2.1.2 Méthode de Neville

Au départ, nous avons N points (x_i, y_i). Nous allons écrire le polynôme $P_{N-1}[x_1, x_2, \ldots, x_N]$ sur les points $\{1, 2, \ldots, N\}$ en fonction des polynômes $P_{N-2}[x_1, x_2, \ldots, x_{N-1}]$ et $P_{N-2}[x_2, x_3, \ldots, x_N]$ sur les points $\{1, 2, \ldots, N-1\}$ et $\{2, 3, \ldots, N\}$.

Nous allons utiliser la formule de Lagrange en partant des points d'interpolation $P_0[x_i]$ de degré 0 pour chaque point i. La courbe associée à chacun de ces polynômes passe par le point (x_i, y_i).

Nous avons les formules suivantes :

$$P_0[x_i](x) = y_i \ \forall i = 1, \ldots, N$$

$$P_1[x_i, x_{i+1}](x) = \frac{(x - x_{i+1})P_0[x_i](x) - (x - x_i)P_0[x_{i+1}](x)}{x_i - x_{i+1}} \ \forall i = 1, \ldots, N$$

$$P_k[x_i, \ldots, x_{i+k}](x) = \frac{(x - x_{i+k})P_{k-1}[x_i, \ldots, x_{i+k-1}](x) - (x - x_i)P_{k-1}[x_{i+1}, x_{i+k}](x)}{x_i - x_{i+k}} \ \forall k = 2, \ldots, N-1,$$

La méthode est la suivante :

— On pose $P_0[x_i](x) = y_i, \forall i = 1, \ldots, N$.
— On utilise les valeurs précédentes 2 à 2 pour calculer $P_1[x_i, x_{i+1}, \forall i = 1, \ldots, N-1$.
— On poursuit le raisonnement jusqu'à $P_{N-1}[x_1, x_2, \ldots, x_N]$.

2.2 Approximation

Nous allons implémenter deux méthodes d'approximation. En effet, nous allons approcher la valeur du polynôme selon une droite de régression pour certains jeux d'essai, et selon un ajustement puissance de type $y = ax^b$ pour d'autres jeux d'essai.

2.2.1 Approximations selon une droite de régression

Nous allons faire l'approximation par une droite représentative du polynôme de degré 1 $f(x) = a_0 + a_1 x$. Notre système d'équation linéaires peut se réécrire :

$$Ax = b \iff \begin{pmatrix} \sum_{i=0}^{n-1} x_i^0 & \sum_{i=0}^{n-1} x_i^1 \\ \sum_{i=0}^{n-1} x_i^1 & \sum_{i=0}^{n-1} x_i^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n-1} y_i x_i^0 \\ \sum_{i=0}^{n-1} y_i x_i^1 \end{pmatrix}$$

En résolvant le système, on obtient les coefficients du polynôme :

$$a_1 = \frac{\bar{xy} - \bar{x}\bar{y}}{\bar{x^2} - (\bar{x})^2}$$

$$a_0 = \frac{\bar{y}\bar{x^2} - \bar{x}\bar{xy}}{\bar{x^2} - (\bar{x})^2}$$

où on a :

$$\bar{x} = \frac{\sum_{i=0}^{n-1} x_i}{n},\ \bar{y} = \frac{\sum_{i=0}^{n-1} y_i}{n},\ \bar{xy} = \frac{\sum_{i=0}^{n-1} x_i y_i}{n},\ \bar{x^2} = \frac{\sum_{i=0}^{n-1} x_i^2}{n}$$

2.2.2 Ajustement puissance

Il suffit de réécrire l'équation $y = ax^b$:

$$y = ax^b \iff ln(y) = ln(ax^b) \iff ln(y) = ln(a) + bln(x) \iff Y = bX + a'$$

avec $Y = ln(y)$, $X = ln(x)$ et $a' = ln(a)$.

Ensuite, il reste à faire l'approximation de la droite obtenue selon la méthode précédente, puis de retrouver les bonnes valeurs en changeant les variables : $y = e^Y$, $x = e^X$ et $a = e^{a'}$.

3 Présentation des programmes commentés

3.1 Structure générale du programme

Le programme est divisé en en six fichiers principaux :

— `approximation.c` : contient l'implémentation de la méthode de régression linéaire et d'approximation ;
— `gestionFichier.c` : contient les fonctions pour la gestion de lecture et d'écriture des points des différents polynômes ;
— `interpolation.c` : contient l'implémentation des polynômes pour les méthodes de Lagrange et de Neville ;
— `main.c` : contient le programme principal faisant appel aux autres fichiers ;
— `trace.py` : contient le programme qui trace les polynômes de Lagrange et Neville (dans le dossier `Trace Python` ;
— `trace_approx.py` : contient le programme qui trace les fonctions d'approximation (dans le dossier `Trace Python` ;

Remarque 3. A chaque fichier `.c` correspond un fichier `.h`, qui contient essentiellement les signatures des fonctions (sauf pour le `main.c`) ;

3.2 Commentaire du programme

Pour en savoir plus sur nos fonctions, il est possible de se référer aux commentaires dans le code (directement dans les fichiers sources). Il est aussi possible de consulter une documentation plus détaillée. Elle se trouve dans le dossier `Documentation` rendu sous le nom de `refman.pdf`.

3.3 Choix d'implémentation

3.3.1 `interpolation.c`

Lagrange Nous avons implémenté de manière « naïve » la fonction permettant de calculer les valeurs du polynôme de Lagrange. En effet, une optimisation de la fonction pour trouver les valeurs du polynôme de Lagrange est plutôt délicate. Ceci s'explique par le fait que si on décide de rajouter un point par lequel le polynôme doit passer, alors on doit recalculer chacun des $li(x)$.

Neville Nous avons pu optimiser le nombre d'opérations en stockant dans un tableau les différentes évaluations polynomiales. Ce tableau est basé sur la méthodes des *différences divisées*. De ce fait, l'ajout d'un point ne recalcule pas tout les polynômes intermédiaires, mais rajoute seulement une ligne et une colonne dans notre tableau, contrairement à la méthode de Lagrange.

3.3.2 `approximation.c`

Pour l'approximation linéaire, nous nous sommes basés sur la méthode exposée dans la section 2.2.1. A la seule différence que pour la valeur a_0, nous avons considéré que $a_0 = \bar{y}_a \times \bar{x}$.

Pour l'approximation d'une équation sous la forme $y = ax^b$, nous nous sommes basés sur la méthode exposée dans la section 2.2.2. Nous avons là aussi considéré le logarithme népérien, puis fait naturellement sa réciproque avec la fonction exponentielle le moment venu.

3.4 Graphe d'appel des fonctions

Voici également le graphe d'appel de nos fonctions en figure 2.

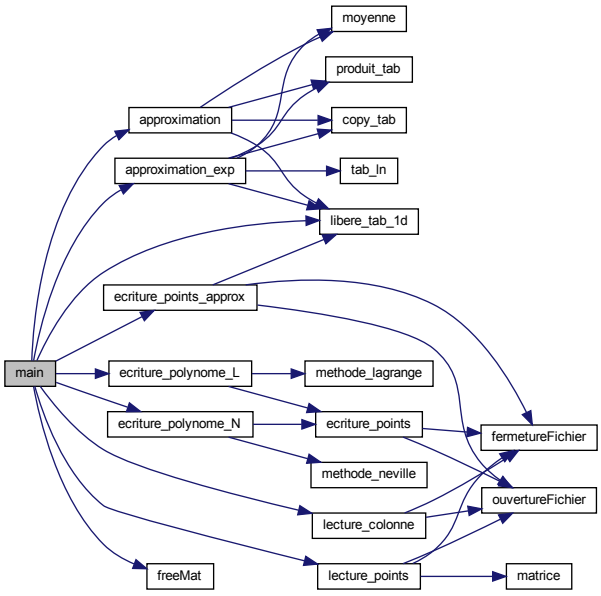


Figure 2 – Graphe d'appel des fonctions

4 Présentation des jeux d'essais et des graphiques

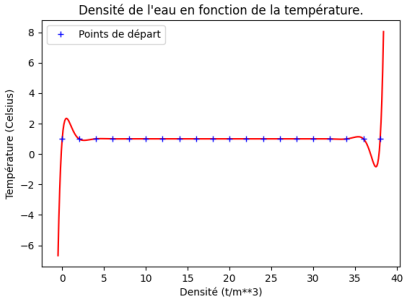
Nous avons à notre disposition quatre jeux d'essais pour tester nos fonctions. Les trois premiers serviront à la fois pour l'interpolation et pour l'approximation, tandis que le quatrième servira uniquement pour l'approximation.

4.1 Densité de l'eau en fonction de la température

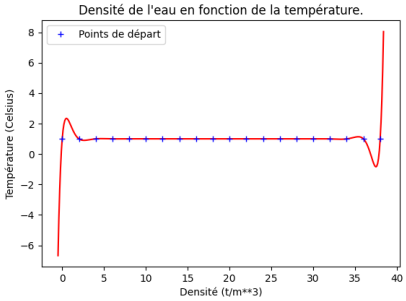
On a à notre disposition les données suivantes :

T (°C)	0	2	4	6	8	10	12	14	16	18
D (t/m^3)	0.99987	0.99997	1.00000	0.99997	0.99988	0.99973	0.99953	0.99927	0.99897	0.99846
T (°C)	20	22	24	26	28	30	32	34	36	38
D (t/m^3)	0.99805	0.999751	0.99705	0.99650	0.99664	0.99533	0.99472	0.99472	0.99333	0.99326

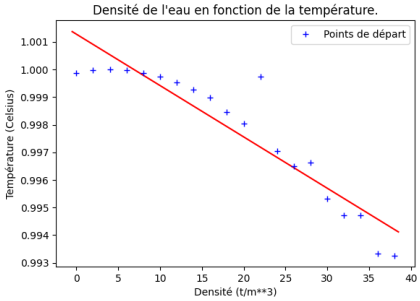
On obtient les graphiques suivants :



(a) Méthode de Lagrange



(b) Méthode de Neville



(c) Droite de régression

Avec la méthode de la droite de régression, le polynôme qui approche au mieux le polynôme recherché est : $-0.000186x + 1.001279$.

4.2 Dépenses mensuelles et revenus

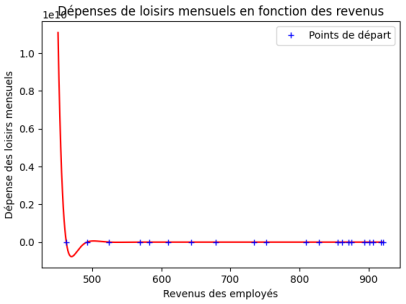
Ici, on s'intéresse à la relation entre les dépenses de loisirs mensuelles (D) et les revenus (R) des employés d'une entreprise.

On a à notre disposition les données suivantes :

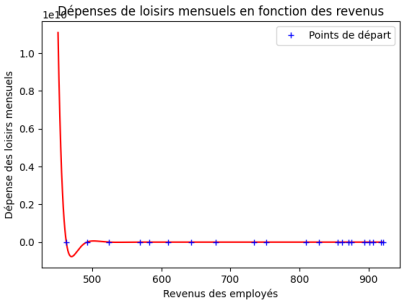
R	752	855	871	734	610	582	921	492	569	462	907
D	85	83	162	79	81	83	281	81	81	80	243

R	643	862	524	679	902	918	828	875	809	894
D	84	84	82	80	226	260	82	186	77	223

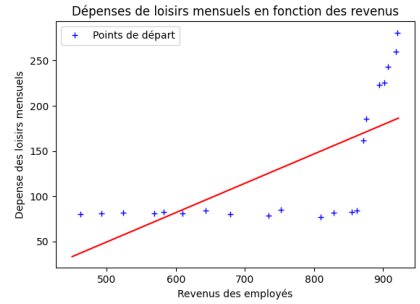
On obtient les graphiques suivants :



(a) Méthode de Lagrange



(b) Méthode de Neville



(c) Droite de régression

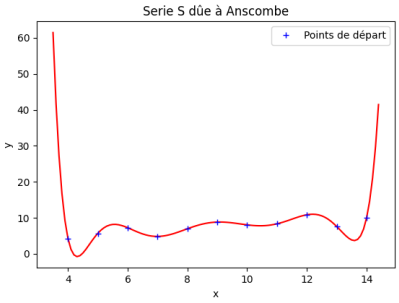
Avec la méthode de la droite de régression, le polynôme qui approche au mieux le polynôme recherché est : $0.324356x - 112.658249$.

4.3 Série S due à Anscombe

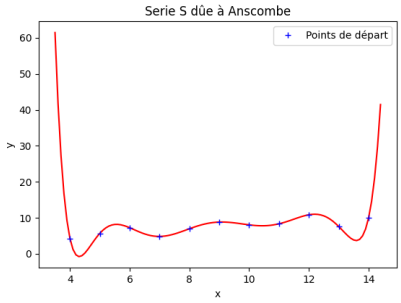
On a à notre disposition les données suivantes :

x_i	10	8	13	9	11	14	6	4	12	7	5
y_i	8.04	6.95	7.58	8.81	8.33	9.96	7.24	4.26	10.84	4.82	5.68

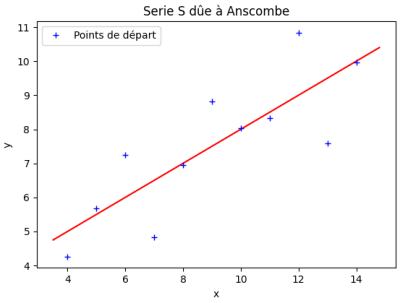
On obtient les graphiques suivants :



(a) Méthode de Lagrange



(b) Méthode de Neville



(c) Droite de régression

Avec la méthode de la droite de régression, le polynôme qui approche au mieux le polynôme recherché est : $0.500092x + 3.000084$.

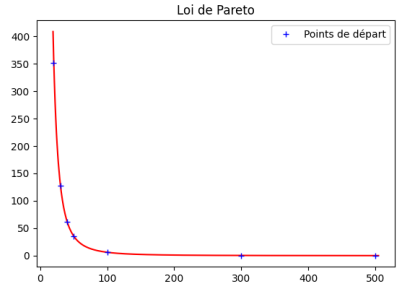
4.4 Vérification de la loi de Pareto (Seulement pour l'approximation)

Théorème 1 (Loi de Pareto). Entre le revenu x et le nombre y de personnes ayant un revenu de type supérieur à x, il existe une relation du type : $y = \frac{A}{x^a} = Ax^{-a}$, où a et A sont des constantes positives caractéristiques de la région considérée et de la période étudiée.

Nous voulons vérifier cette loi pour les données suivantes :

x_i	20	30	40	50	100	300	500
y_i	352	128	62.3	35.7	6.3	0.4	0.1

On obtient, pour l'approximation, le graphique suivant :



(a) Ajustement puissance

Avec la méthode de l'ajustement puissance, le polynôme qui approche au mieux le polynôme recherché est : $696613.625000x^{-2.527193}$. C'est à dire, nous avons trouvé $A = 696613.625000$ et $a = 2.527193$

5 Commentaires sur les jeux d'essais

5.1 Interpolation

5.1.1 Unicité du polynôme

Pour les trois jeux d'essais testés, les deux polynômes obtenus par les méthodes de Lagrange et de Neville passent par tous les points (x_i, y_i) de départ. Cela s'observe dans les fichiers obtenus lors de l'exécution du programme.

Or, par le théorème de l'*unisolvance*, il existe un unique polynôme passant par un échantillon de points. Donc, les deux polynômes obtenus sont identiques.

Cependant, des erreurs de calcul liées à l'encodage machine des nombres flottants font que les deux polynômes ne sont pas exactement identiques. Cette différence est faible et peut être négligée dans la plupart des cas en fonction de la précision voulue.

5.1.2 Adéquation du polynôme aux jeux d'essais

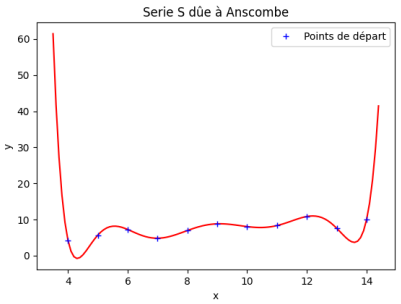
Comme expliqué plus haut, avec les deux méthodes et avec les trois jeux d'essais testés, les polynômes obtenus passent bien par les points de départ. Ils sont donc en adéquation avec les jeux d'essais.

5.1.3 Influence de la modification d'un ou plusieurs points sur le polynôme

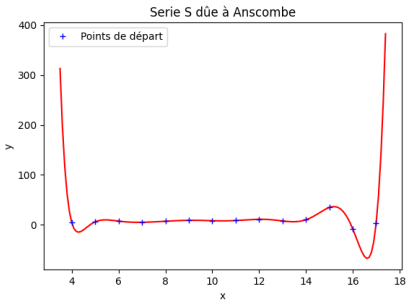
Nous avons testé plusieurs modifications sur les jeux d'essais. Nous avons choisi le jeu d'essai présenté en 4.3 car c'est celui où les changements sont les plus visibles.

Nous avons d'abord rajouté les points (15, 34.51), (16, -8.54) et (17, 3.141578). Puis, nous avons modifié le point d'abscisse 14 pour qu'il prenne les valeurs (14, -3.50).

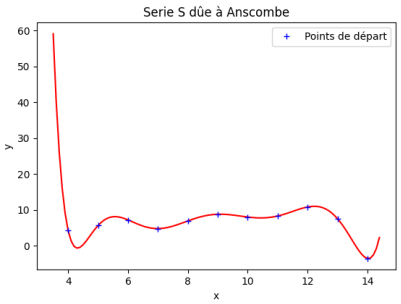
On obtient les graphiques ci-dessous :



(a) Polynôme de base



(b) Ajout de points



(c) Modification d'un point

5.1.4 Évaluation des coûts en place et en temps

Voici un tableau résumant les temps d'exécution en secondes en fonction des jeux d'essais et de la méthode choisie :

Jeu d'essai / Méthode	Lagrange	Neville	Approximation
Densité de l'eau en fonction de la température	0.005959	0.004214	0.000058
Dépenses mensuelles et revenus	0.043813	0.043887	0.000031
Série due à Anscombe	0.000840	0.000849	0.000028
Vérification de la loi de Pareto	–	–	0.000052

En ce qui concerne les méthodes d'interpolation, on remarque que la méthode de Neville est plus rapide que celle de Lagrange pour les jeux d'essais avec beaucoup de points (Jeu 1). En revanche, pour un nombre limité de points (Jeux 2 et 3), c'est la méthode de Lagrange qui est plus rapide.

5.2 Approximation

On observe que pour l'approximation, les temps d'exécution sont plus rapides que ceux de l'interpolation quelque soit le jeu d'essai choisi. En revanche, on observe que le polynôme obtenu ne passe pas par tous les points. Il est beaucoup moins précis que celui obtenu par interpolation.

Dans le cas du jeu d'essai 4.4, on observe en revanche que le polynôme obtenu semble passer par tous les points. On a obtenu : $696613.625000x^{-2.527193}$ On a donc une relation du type : $y = \frac{A}{x^a} = Ax^{-a}$, où a et A sont des constantes positives. La loi de Pareto est donc vérifiée pour ce jeu d'essai.

6 Conclusion générale

Pour conclure, les trois méthodes (Lagrange, Neville et droite de régression) permettent de trouver l'équation d'un polynôme qui passe par plusieurs points donnés. Les méthodes basées sur l'interpolation donnent le polynôme exact, mais il n'est pas forcément trouvé en un temps raisonnable, surtout s'il y a beaucoup de points. On remarque tout de même que la méthode de Lagrange est plus rapide quand il y a beaucoup de points. Sinon,

il est préférable d'utiliser la méthode de Neville. Les méthodes d'approximation sont beaucoup plus rapides que les méthodes d'interpolation, mais donnent des polynômes moins précis.

Il est donc préférable d'utiliser les méthodes d'approximation lorsque l'on souhaite avoir une idée rapide de l'évolution de la fonction, et d'utiliser les méthodes d'interpolation pour trouver des valeurs de manière précise. Le choix de la méthode dépendra du nombre de points à traiter.