# TP Interpolation polynômiale et Approximation.

Salmân Abibou & Rodrigo Ferreira Rodrigues
Université Clermont Auvergne

November 7, 2022

# Contents

# 1 Rappel des méthodes

On notera $N$ le nombre de points $(x_i, y_i)$ connues.

## 1.1 Méthodes d'interpolation

### 1.1.1 Lagrange

On cherche à exprimer le polynôme $P_{N-1}$ de la forme :

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

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

$$l_i(x) = \prod_{j=0, j\neq i}^{(N-1)} \frac{x - x_j}{x_i - x_j} \tag{2}$$

### 1.1.2 Neville

Le but de la méthode est d'obtenir le polynôme $P_{N-1}[x_1, x_2, ..., x_N]$ en fonction des deux polynômes de degré inférieur : $P_{N-2}[x_1, x_2, ..., x_{N-1}]$ et $P_{N-2}[x_2, x_3..., x_N]$. Ces deux polynômes se trouvent chacun grâce à deux autres polynômes de degré inférieur, et ainsi de suite...
Ainsi en partant des polynômes d'interpolation $P_0[x_i]$, on arrive à $P_{N-1}$.
Pour cela, on a les formules :

$$\begin{aligned}
&\forall x, P_0[x_i] = y_i, i = 1, ..., N \\
&\forall x, P_1[x_i, xi+1](x) = \frac{(x - x_{i+k})P_0[x_i](x) + (x_i - x)P_0[x_{i+1}](x)}{x_i - x_{i+1}}, i = 1, ..., N \\
&\forall x, P_k[x_i, ..., x_{i+1}](x) = \frac{(x - x_{i+k})P_{k-1}[x_i, ..., x_{i+k-1}](x) + (x_i - x)P_{k-1}[x_{i+1}, ..., xi+k](x)}{x_i - x_{i+k}}, \\
&\forall k = 2, ..., N-1
\end{aligned} \tag{3}$$

## 1.2 Méthodes d'approximation

L'approximation consiste à obtenir une fonction $f(x)$ dont la distance moyenne entre les $N$ points connues et la courbe soit minimum.

### 1.2.1 Droite de régression

Le but est d'approcher la fonction $f(x)$ grâce à une droite. On cherche ainsi les coefficients $a_0$ et $a_1$ tel que $f(x) = a_0 + a_1 x$.
A partir de système d'équation :

$$\begin{bmatrix} \sum_{i=0}^{N-1} x_i^0 & \sum_{i=0}^{N-1} x_i^1 \\ \sum_{i=0}^{N-1} x_i^1 & \sum_{i=0}^{N-1} x_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N-1} y_i x_i^0 \\ \sum_{i=0}^{N-1} y_i x_i^1 \end{bmatrix}$$

On obtient les coefficients :

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

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

### 1.2.2 Ajustement puissance du type $y = ax^b$

Le but de cette méthode est aussi d'obtenir une fonction, approchant au mieux les $N$ points connues.
On cherche alors $f(x) = ax^b$.

Pour cela, on manipule cette équation :

$$\begin{aligned} y &= ax^b \\ log(y) &= log(ax^b) \\ log(y) &= log(a) + b \cdot log(x) \end{aligned}$$

On pose :

$$Y = log(y_i)$$
$$X = log(x_i)$$
$$A = log(a)$$

On obtient alors :

$$Y = A + b \cdot X$$
$$A = \overline{Y} - b \cdot \overline{X}$$
$$a = 10^A$$

avec :

$$b = \frac{\overline{yx} - \bar{x}\bar{y}}{\bar{x}^2 - (\bar{x})^2}$$

# 2 Explication fonctions du programme

## 2.1 Fonction Cardinale

```
/*fonction permettant le calcul de la fonction auxiliaire cardinale
*/
double cardinale(int N, int i, double *val, double x){  // Soit N le nombre de points, i l'indice, val le tableau contenant les valeurs de x_i, et x le réel rentré par l'utilisateur
    double denominateur=1;
    double numerateur=1;
    for(int j=0; j<N; j++){
        if(j!=i){   //pour j=0 allant à N-1 (avec j!=i), on calcule séparément le numérateur et le dénominateur
            denominateur*=(val[i]- val[j]); //on multiplie la valeur du dénominateur possédée par x_i-x_j
            numerateur*=(x-val[j]); //on multiplie la valeur du numénateur possédé par x-x_j
        }
    }
```

```
12      return numerateur/denominateur; //on renvoie le résultat
        approché de la fraction
13  }
```

Listing 1: Fonction auxiliaire cardinale.

Cette fonction a pour rôle de calculer la fonction cardinale à un indice $i$ et pour une valeur $x$ rentré au préalable par l'utilisateur. (voir section 2)
Cette fonction est auxiliaire à celle de Lagrange.

**Entrée** : N, le nombre de points, le réel x, un indice i et un tableau contenant les valeurs x du jeu
**Sortie** : $l_i(x)$
**Complexité spatiale** : $O(n)$
**Complexité temporelle** : $O(n)$

## 2.2 Fonction Lagrange

```
1  double lagrange(int N, double **val, double x){
2      double y=0;
3      for(int i=0;i<N;i++){
4          y+=val[1][i]*cardinale(N,i,val[0],x);
5      }
6      return y;
7  }
```

Listing 2: Fonction Lagrange.

Cette fonction permet d'obtenir une solution $y$ selon la variable $x$ rentrée par l'utilisateur et les points données.
Pour cela, la fonction utilise la méthode d'interpolation de Lagrange afin d'obtenir un polynôme $P_{N-1}$. (voir section 1)

**Entrée** : N, le nombre de points, le réel x et un tableau contenant les valeurs x et y du jeu
**Sortie** : $y$
**Complexité spatiale** : $O(n^2)$
**Complexité temporelle** :$O(n^2)$

## 2.3 Fonction Neville

```
double neville(int i, int k, double x, double **val){

    if(k==0){ //quand le polynôme atteint le degré 0 on renvoie y_i
        return val[1][i];
    }
    else{
        return(((x-val[0][i+k])*neville(i, k-1, x, val) + (val[0][i] - x)*neville(i+1, k-1,x, val))/(val[0][i]-val[0][i+k]));
    }
}
```

Listing 3: Fonction Neville.

Il s'agit d'une fonction récursive simulant la méthode d'interpolation de Neville. Le cran d'arrêt va être la variable $k$ indiquant le degré du polynôme. Quant elle atteint zéro, on renvoie la valeur $y_i$. Le $y$, solution recherché est ensuite calculé grâce aux formules données. (voir section 1.1.2)

**Entrée** : N, le nombre de points, le réel x, un indice i et un tableau contenant les valeurs x et y du jeu
**Sortie** : $y$
**Complexité spatiale** : $O(1)$
**Complexité temporelle** : $O(log(n))$

## 2.4 Fonction régression

```
double *regression(double **val, int n){
    double *fonct=malloc(sizeof(double)*2); //permet de stocker les  coefficients de la droite de regression a0 et a1
    double moy_x=0,moy_y=0,moy_xy=0,moy_x_au_carre=0; //initialisation des variables
    for(int i=0;i<n;i++){
        moy_x_au_carre+=powf(val[0][i],2); //la somme des carrés des abscisses(x²_i)
        moy_y+=val[1][i]; // la somme des ordonnées(y_i)
        moy_x+=val[0][i]; // la somme des abscisses(x_i)
        moy_xy+=val[0][i]*val[1][i]; // la somme du produit des ordonnées et des abscisses(xy_i)
```

```
    }

    moy_y=moy_y/n; //la moyenne des y_i
    moy_x_au_carre=moy_x_au_carre/n; //la moyenne des x^2_i
    moy_x=moy_x/n; //la moyenne des x_i
    moy_xy=moy_xy/n; //la moyenne des xy_i

    fonct[1]=(moy_xy - moy_x*moy_y)/(moy_x_au_carre-powf(moy_x,2)); //coefficient a1
    fonct[0]=(moy_y*moy_x_au_carre-moy_x*moy_xy)/(moy_x_au_carre-powf(moy_x,2)); //coefficient a0
    return fonct;

}
```

Listing 4: Fonction de la droite de regression linéaire.

La fonction **régression** permet d'obtenir les coefficients $a_0$ et $a_1$ de la droite de régression. (voir section 1.2.1)

**Entrée** : N, le nombre de points et un tableau contenant les valeurs x et y du jeu
**Sortie** : $a_1$ et $a_0$, les coefficients de la fonction recherchée, stockées dans un tableau
**Complexité spatiale** : $O(n)$
**Complexité temporelle** : $O(n)$

## 2.5 Fonction axb

```
double *axb(double **val, int n){
    double *fonct=malloc(sizeof(double)*2);//permet de stocker les  coefficients de l'ajustement puissance a et A
    double moy_x=0,moy_y=0,moy_xy=0,moy_x_au_carre=0;//initialisation des variables
    for(int i=0;i<n;i++){
        moy_x_au_carre+=powf(log10f(val[0][i]),2);
        moy_y+=log10f(val[1][i]);
        moy_x+=log10f(val[0][i]);
        moy_xy+=log10f(val[0][i])*log10f(val[1][i]);
    }
    moy_y=moy_y/n;
    moy_x_au_carre=moy_x_au_carre/n;
```

```
12      moy_x=moy_x/n;
13      moy_xy=moy_xy/n;
14      double b=(moy_xy - moy_x*moy_y)/(moy_x_au_carre-powf(moy_x,2)
        ); //calcul du coeff b
15      double a=powf(10, moy_y-(b*moy_x)); //calcul du coeff a
16      fonct[0]=a;
17      fonct[1]=b;
18      return fonct;
19
20  }
```

Listing 5: Fonction de l'ajustement puissance.

Cette fonction permet le calcul des coefficients $a$ et $b$ d'une fonction du type $ax^b$. (voir section 1.2.2)

**Entrée** : N, le nombre de points et un tableau contenant les valeurs x et y du jeu
**Sortie** : $a$ et $b$, les coefficients de la fonction recherchée, stockées dans un tableau
**Complexité spatiale** : $O(n)$
**Complexité temporelle** : $O(n)$

# 3 Analyse jeux d'essais

## 3.1 Densité ($D$) de l'eau en fonction de la température (T)

Les données connues représentent la densité de l'eau (en $t/m^3$) en fonction de la température (en $°C$).
On remarque avec ces données que la densité diminue alors que la température augmente.
Cette série contient 20 points (en rouge sur le graphique) avec une moyenne des abscisses de 19 et une moyenne d'ordonnées de 0.997638.

On a déterminé les $y_i$ de chacun des $x_i$ allant de 0 à 40 par les polynômes obtenus respectivement par la méthode de Lagrange et de Neville et on les a représentés sur les graphiques ci-dessous.

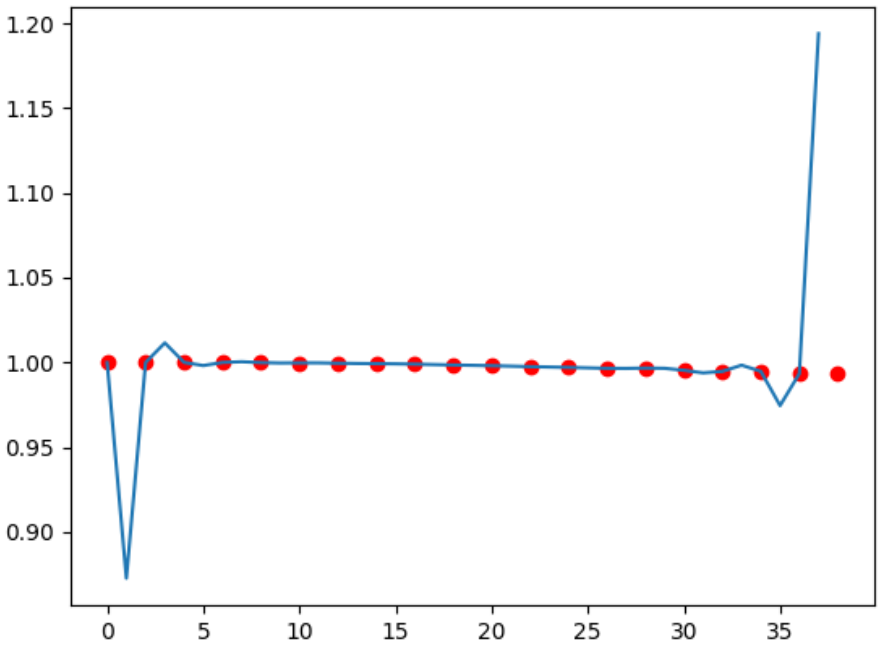


Figure 1: Interpolation de Lagrange pour le jeu d'essai 1.

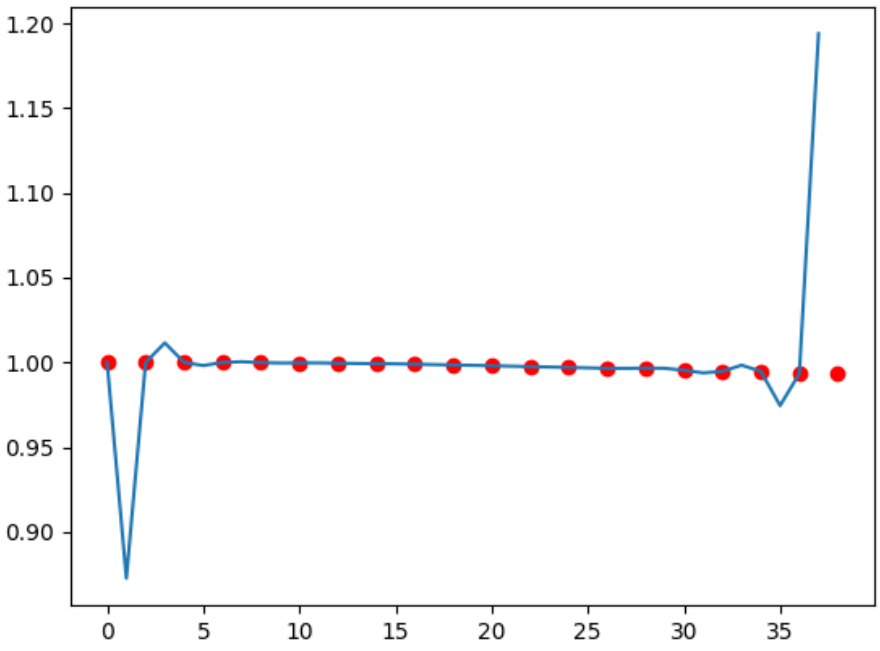


Figure 2: Interpolation de Neville pour le jeu d'essai 1.

On remarque qu'on obtient les mêmes graphiques pour les méthodes de Neville et de Lagrange traduisant l'unicité des polynômes.
On remarque qu'à partir de $x = 36$ le polynôme croit de façon exponentielle. Le polynôme n'est plus fiable.

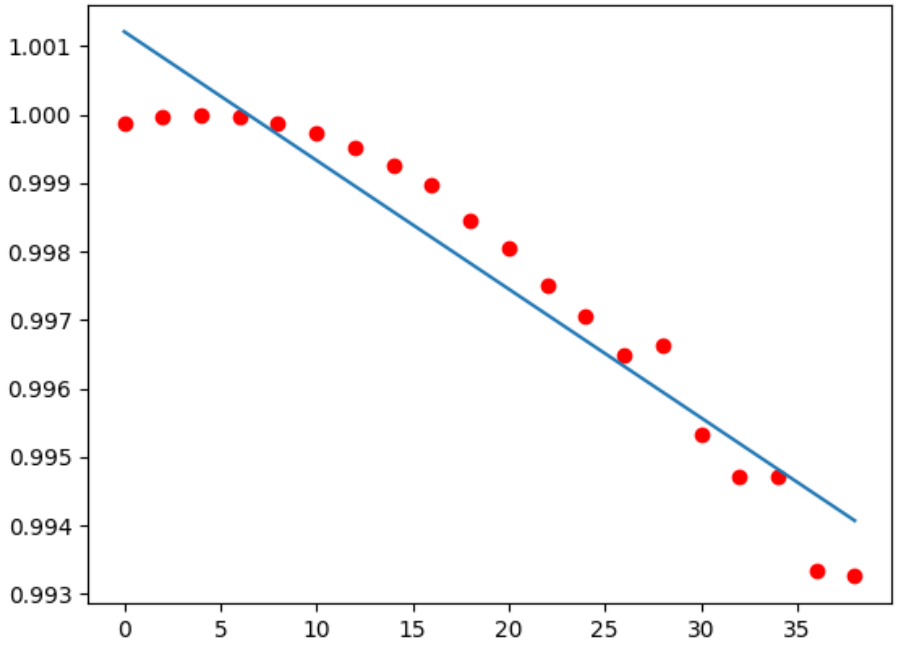


Figure 3: Approximation selon la régression linéaire pour le jeu d'essai 1.

La droite de régression décroît et semble bien représenter la situation.

## 3.2 Dépenses mensuelles et revenus

Cette série établit le lien entre les revenus des employés d'une entreprise et leurs dépenses.
On remarque que les dépenses sont un peu près les mêmes pour les employés ayant un revenu inférieur à 900. Après cela, les dépenses augmentent considérablement.

Cette série contient 21 points (en rouge sur le graphique) avec une moyenne des abscisses de 747.095238 et une moyenne d'ordonnées de 129.666667.

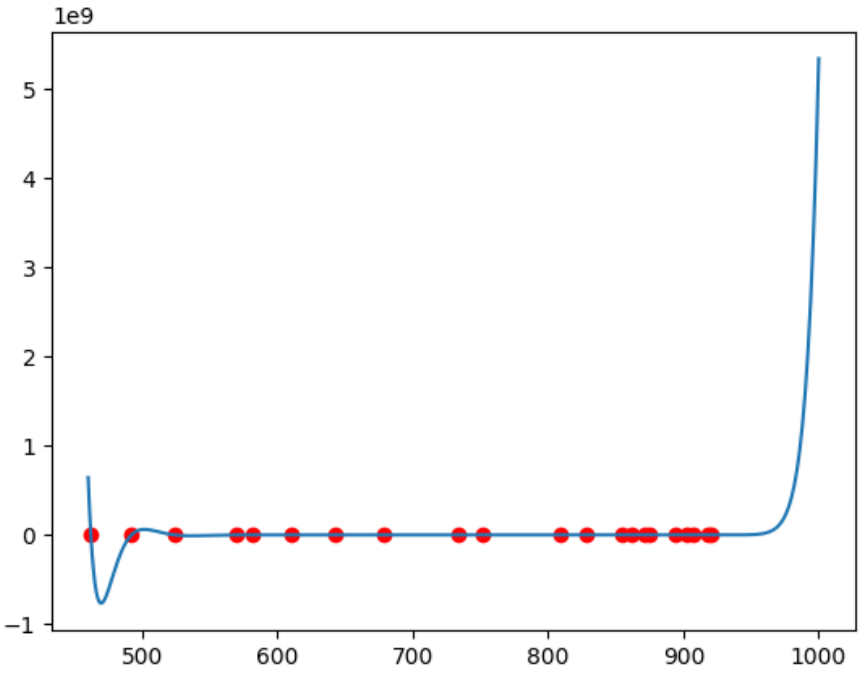


Figure 4: Interpolation de Lagrange pour le jeu d'essai 2.

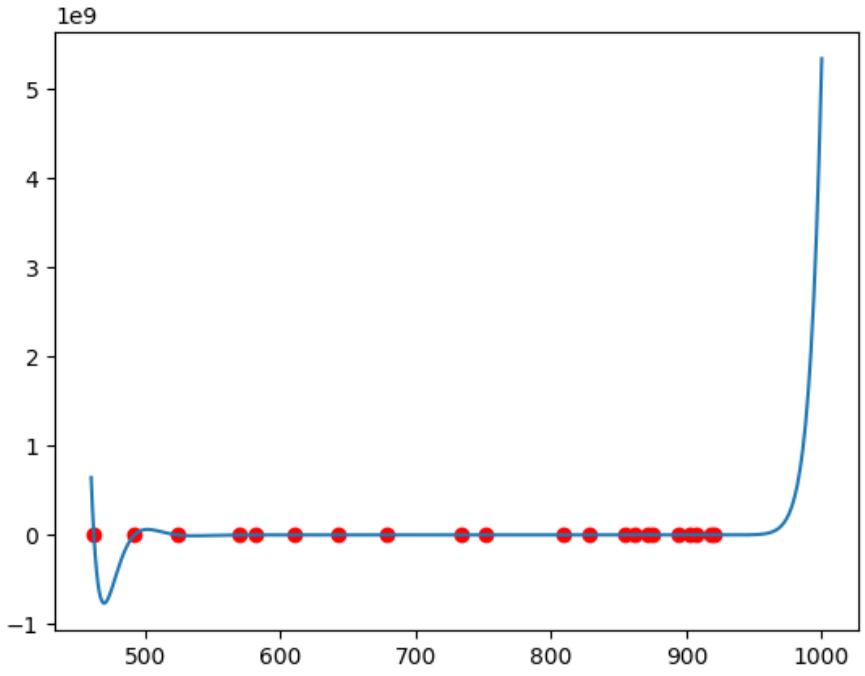


Figure 5: Interpolation de Neville pour le jeu d'essai 2.

On remarque encore une fois l'unicité des polynômes.
De nouveau, si on s'éloigne du jeu de données les valeurs renvoyées par les polynômes ne sont plus fiables.

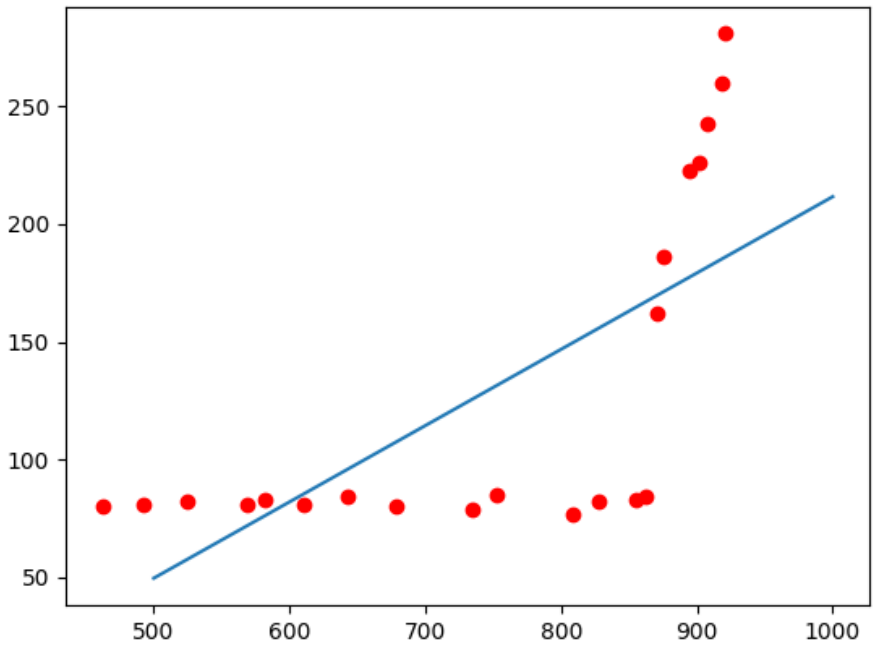


Figure 6: Approximation selon la régression linéaire pour le jeu d'essai 2.

L'approximation semble appropriée à ce cas car on elle démontre bien la tendance d'augmentation des dépenses avec le revenue.

## 3.3 Série $S$ dûe à Ascombe

Cette série fait partie du quartet d'Anscombe qui est constitué de quatre ensembles de données qui ont les mêmes propriétés statistiques simples. Ils ont été construits par le statisticien Francis Anscombe dans le but de démontrer l'importance de tracer les graphiques avant d'analyser des données, car cela permet notamment d'estimer l'incidence des données aberrantes sur les différentes indices statistiques que l'on pourrait calculer[1].

[1] `https://fr.wikipedia.org/wiki/Quartet_d%27Anscombe`

Cette série contient 11 points de données (représentés en point rouges) avec une moyenne des abscisses de 9 et une moyenne des ordonnées de 7.5.

Ici on a déterminé les $y_i$ de chacun des $x_i$ allant de 0 à 20 par les polynômes obtenus respectivement par la méthode de Lagrange et de Neville et on les a représentés sur les graphiques ci-dessous.

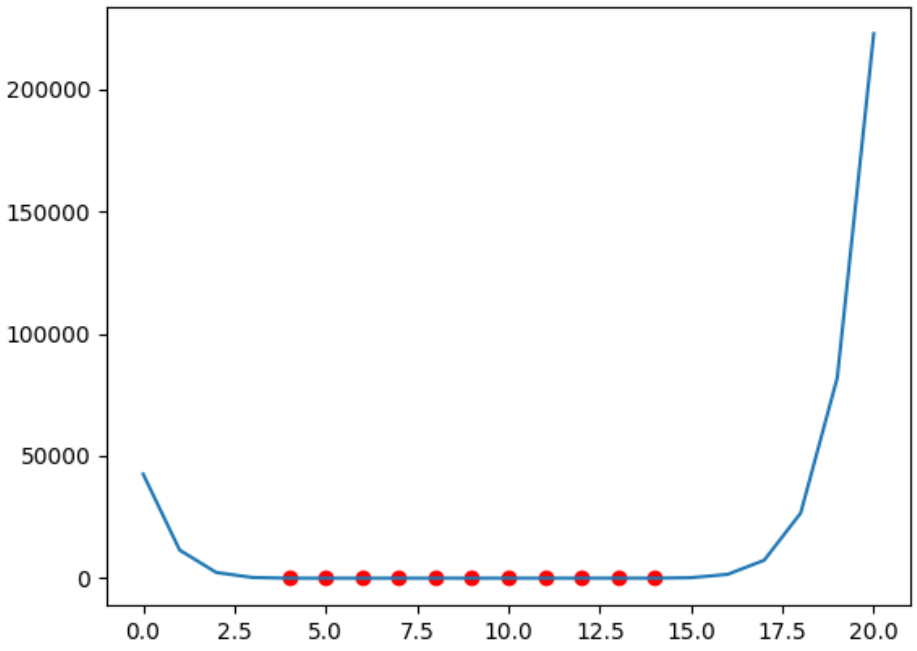


Figure 7: Interpolation de Lagrange pour le jeu d'essai 3.

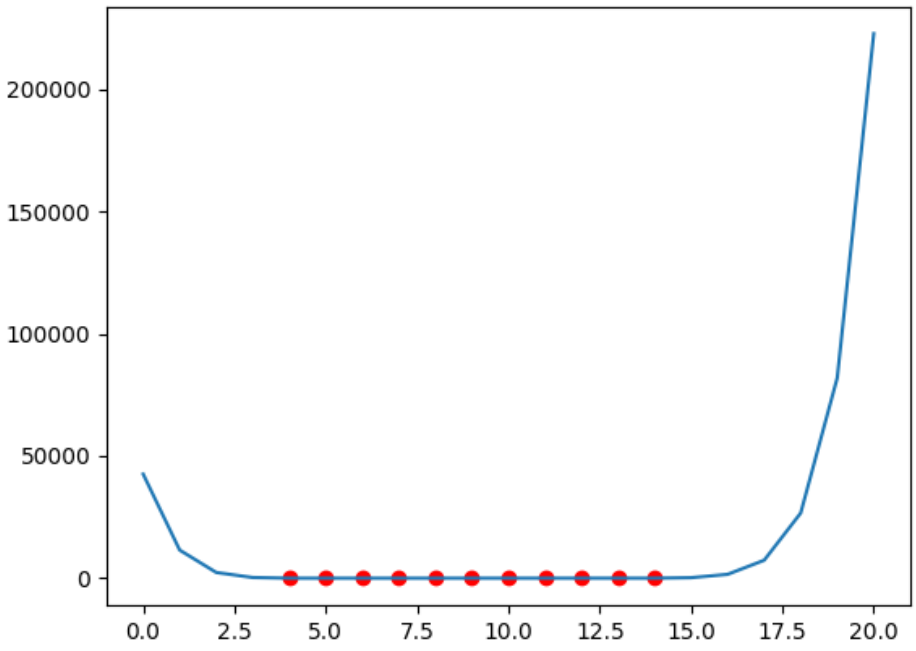


Figure 8: Interpolation de Neville pour le jeu d'essai 3.

Les polynômes d'interpolation de Lagrange et de Neville n'approchent pas bien l'ensemble de données là où les données ne sont pas connues.
En fait, lorsqu'on augmente le nombre de points, on constate que le polynôme se met à osciller fortement entre les points $x_i$ avec une amplitude de plus en plus grande, comme l'illustre les figures ci-dessus.

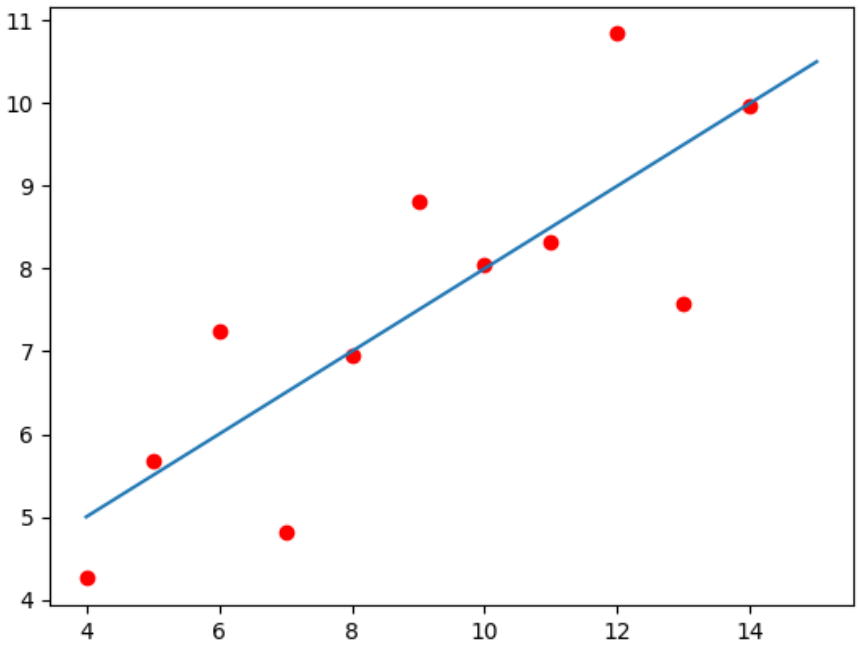


Figure 9: Approximation selon la régression linéaire pour le jeu d'essai 3.

Le but de l'approximation par la droite de régression est de trouver la droite qui passe par le plus de points de l'ensemble de données.

## 3.4 Loi de Pareto

On vérifie la loi de Pareto entre un revenu $x$ et le nombre de personnes $y$ ayant un revenu supérieur à $x$. On remarque après calcul qu'un pourcentage cumulé de 82.08% ont un revenu supérieur à 50.

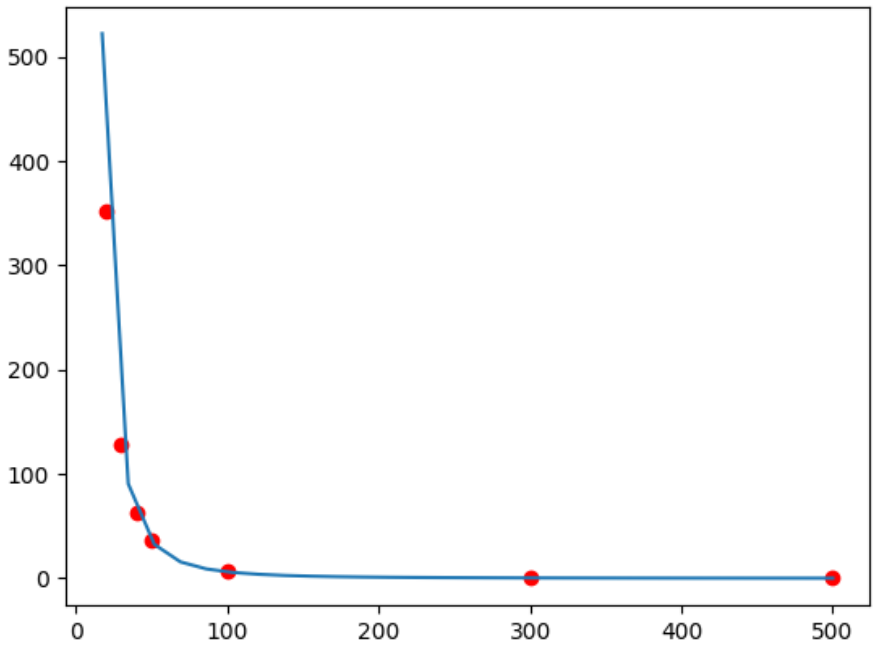


Figure 10: Ajustement puissance pour le jeu d'essai 4.

Plus le revenu $x$ augmente, plus vite le nombre de personnes $y$ ayant un revenu supérieur à $x$ diminue de façon exponentielle et tend vers 0.

# 4 Conclusion

Les méthodes d'interpolation de Lagrange et de Neville permettent d'obtenir un même polynôme (ce qui démontre l'unicité du polynôme)pour un même jeu de données.

De plus, on remarque la représenattion du polynôme sur les graphiques est en parfaite adéquation avec les données connues et quelque fois en inadéquation avec les données non connues. Après modification (ajout de plusieurs points de données) des jeux de données, on a remarqué une fluctuation avec de grandes amplitudes surtout au niveau des points de données aux deux extrémités

du jeu de données.

Pour approcher une fonction avec des polynômes, on peut préférer utiliser des polynômes par morceaux. Dans ce cas, pour améliorer l'interpolation, on augmente le nombre de morceaux et non le degré des polynômes (Segmentation) ce qui peut corriger les oscillations de grandes amplitudes remarquées dans les graphiques.

Lorsque la relation entre les $x$ et $y$ ne sont pas reliés par une formule alors on peut préférer l'approximation qui permet de libérer une tendance de ces données (par exemple le lien entre les dépenses et les revenus 3.2).