

---

# Compte rendu TD1 - Interpolation et approximation

Celine Bouby

Compte rendu TD1

Réalisé par Juliette BLUEM & Axel  
THOUVENIN

13 juin 2021

# Table des matières

|          |                                       |          |
|----------|---------------------------------------|----------|
| <b>1</b> | <b>Exercice n°1</b>                   | <b>3</b> |
| 1.1      | Introduction . . . . .                | 3        |
| 1.2      | Lagrange Newton . . . . .             | 3        |
| 1.2.1    | Code Matlab . . . . .                 | 3        |
| 1.2.2    | Courbes obtenues . . . . .            | 4        |
| 1.3      | Splines cubiques . . . . .            | 5        |
| 1.3.1    | Code Matlab . . . . .                 | 5        |
| 1.3.2    | Courbes obtenues . . . . .            | 5        |
| 1.4      | Modèle linéaires . . . . .            | 6        |
| 1.4.1    | Code Matlab . . . . .                 | 6        |
| 1.4.2    | Courbes obtenues . . . . .            | 6        |
| 1.5      | Modèles polynomiales . . . . .        | 7        |
| 1.5.1    | Code Matlab . . . . .                 | 7        |
| 1.5.2    | Courbes obtenues . . . . .            | 7        |
| 1.6      | Conclusion . . . . .                  | 8        |
| <b>2</b> | <b>Exercice n°2</b>                   | <b>9</b> |
| 2.1      | Introduction . . . . .                | 9        |
| 2.2      | Approximation exponentielle . . . . . | 9        |
| 2.3      | Comparaison . . . . .                 | 10       |

# 1 Exercice n°1

## 1.1 Introduction

Dans cet exercice, nous allons comparer les méthodes d'interpolation et d'approximation vues en cours, à savoir : Lagrange, Newton, splines cubiques, modèles linéaires et polynomiales.

Nous appliquerons ces méthodes à l'étalonnage d'un instrument. Nous disposons d'un tableau comparant des valeurs mesurées par l'instrument et les valeurs exactes associées.

## 1.2 Lagrange Newton

### 1.2.1 Code Matlab

```
%LAGRANGE = NEWTON !! 6pts donc d°5
LagNew = polyfit(mesuree, reelle, 5);
xi = 0.5030:0.01:2.4649; %de 0.5 à 2.47 pp de 0.01
yplot = polyval(LagNew, xi);

plot(mesuree, reelle, 'square', xi, yplot);
```

FIGURE 1.1 – Code pour interpolation avec les méthodes de Lagrange et Newton

La methode "polyfit(x,y,n)" permet de générer un polynôme de degrés n. Dans ce cas on travail sur une interpollation de Lagrange - Newton. Ce qui veut dire que le degrés dépend directement du nombre de points. Ici 6points donc d n°5.

La variable xi permet d'augmenter le nombre de points à afficher.

La fonction "polyval(polynome,xi)" permet de générer les points de notre polynôme aux valeurs données.

### 1.2.2 Courbes obtenues

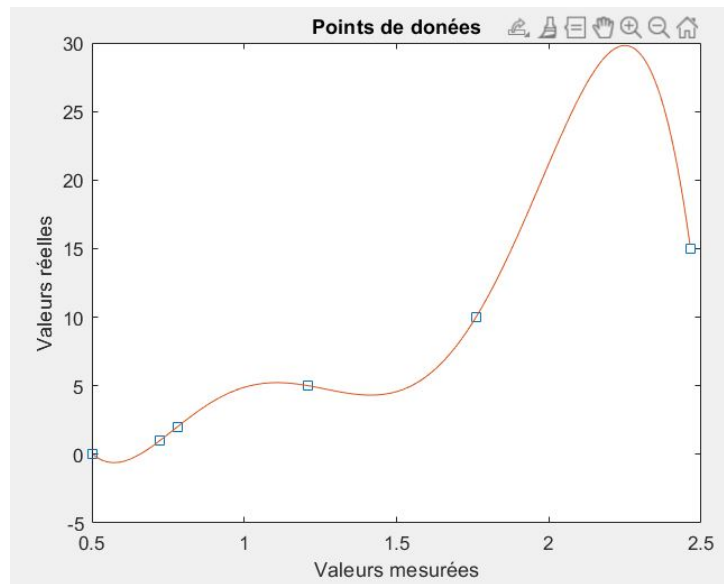


FIGURE 1.2 – Courbes obtenues avec les méthodes de Lagrange et Newton

La courbe générée passe par les six points mesurés. En effet, on utilise un polynôme d'interpolation de degrés cinq. Cependant, on remarque des effets de bords. C'est le principal défaut de ces méthodes. Plus le nombre de points est grand, plus le degrés du polynôme est élevé et plus les effets de bords sont importants.

## 1.3 Splines cubiques

### 1.3.1 Code Matlab

```
%INTERPOLATION PAR SPLINES CUBIQUES  
Spline = spline(mesuree, reelle);  
ysplot = ppval(Spline, xi);  
  
plot(mesuree, reelle, 'square', xi, yplot, xi, ysplot);
```

FIGURE 1.3 – Code pour interpolation avec la méthode des splines cubiques

La fonction "spline(x,y)" permet de générer un spline cubique interpolant y en fonction de x. La fonction "ppval(ys,xi)" permet d'évaluer un polynôme par morceau.

### 1.3.2 Courbes obtenues

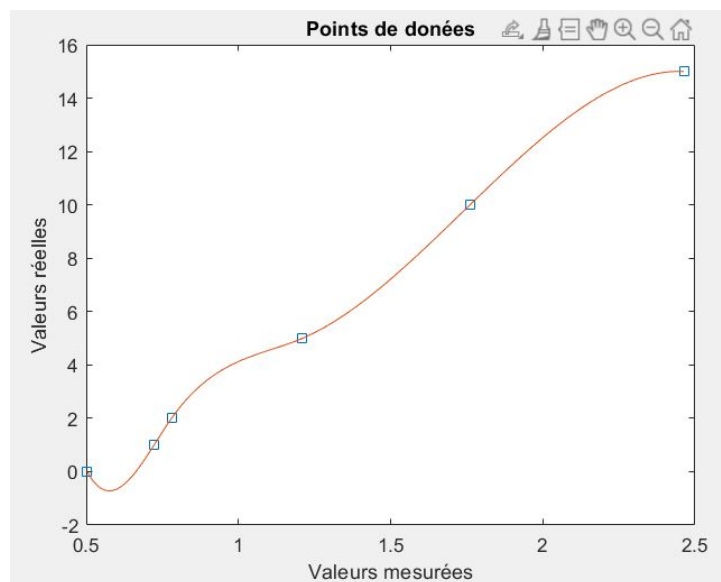


FIGURE 1.4 – Courbes obtenues avec la méthode des splines cubiques

La courbe générée passe par les six points mesurés. En effet, on utilise une méthode d'interpolation.

Grâce à cette méthode, les effets de bords sont considérablement réduits.

Cependant, l'allure de la courbe obtenue dépend des intervalles entre chaque points de mesure.

## 1.4 Modèle linéaires

### 1.4.1 Code Matlab

```
%APPROXIMATION de la forme  $Ax + b$  donc d°1  
AppLin = polyfit(mesuree, reelle, 1);  
yplot = polyval(AppLin, xi);
```

FIGURE 1.5 – Code pour approximation avec un modèle linéaire

On génère une approximation de la forme linéaire donc de degrés 1.

### 1.4.2 Courbes obtenues

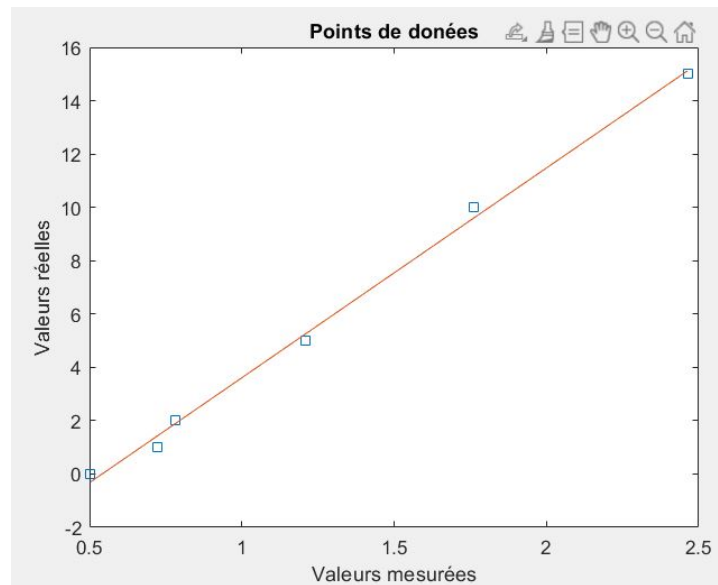


FIGURE 1.6 – Courbes obtenues avec une approximation via un modèle linéaire

La courbe obtenue est linéaire. De ce fait, les effets de bords sont inexistant. Seulement, elle ne passe pas pas l'intégralité des points de mesures.

## 1.5 Modèles polynomiales

### 1.5.1 Code Matlab

```
%POLYNOMIALE de la forme  $Ax^2 + bx + c$  donc d°2  
AppPol = polyfit(mesuree, reelle, 2);  
ypolplot = polyval(AppPol, xi);
```

FIGURE 1.7 – Code pour approximation avec un modèle polynomiale

On génère une approximation de la forme linéaire donc de degrés supérieur à 1.

### 1.5.2 Courbes obtenues

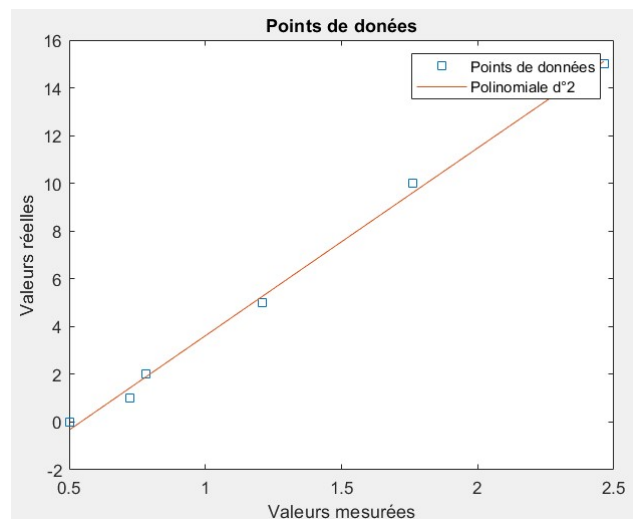


FIGURE 1.8 – Courbes obtenues avec une approximation via un modèle polynomiale

La courbe d'approximation obtenue grâce à cette méthode tend à se rapprocher d'une courbe linéaire.

Elle ne passe pas exactement par les points de mesures.

## 1.6 Conclusion

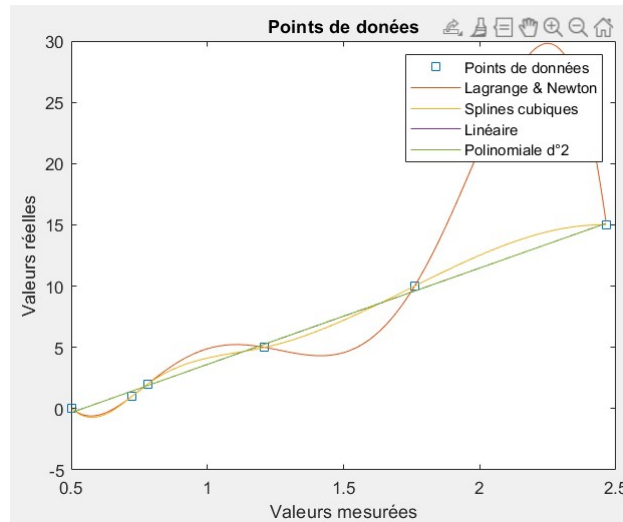


FIGURE 1.9 – Comparaison graphique des différentes courbes étudiées précédemment

Après analyse, nous privilégierons une méthode d'approximation linéaire. En effet, les courbes d'étalonnage ont clairement une tendance à la linéarité.

Bien que notre approximation n'est pas exacte (elle ne passe pas par les points de mesure), elle est ici largement suffisante.



## 2 Exercice n°2

### 2.1 Introduction

Dans cet exercice, nous allons étudier l'approximation exponentielle. Pour cela, nous disposons de données sur la pression atmosphérique pour des altitudes de 0 à 1000m au dessus du niveau de la mer. Notre but sera tout d'abord d'établir une courbe d'approximation avec nos données, puis d'extrapoler nos résultats à des altitudes supérieures.

### 2.2 Approximation exponentielle

```
% Recuperation des données
donnees = importdata('Exercice2');
altitude = donnees(:,1);
pression = donnees(:,2);

% Changement de variables
logPression = log(pression);
% Creation du polynome
P = polyfit(altitude,logPression,1)
a = exp(P(2))
k = P(1)

% Nombre de points tracé
altiplot = [0:1:1000];
% Approximation exponentielle
pressplot = a*exp(k*altiplot);
% Figure
figure
plot(altiplot,pressplot,altitude,pression, '*');
```

FIGURE 2.1 – Code pour l'approximation exponentielle

Les étapes phares de notre code sont indiquées en commentaire de ce dernier. Nous obtenons :

$$a = 1014.8$$

$$k = 12487$$

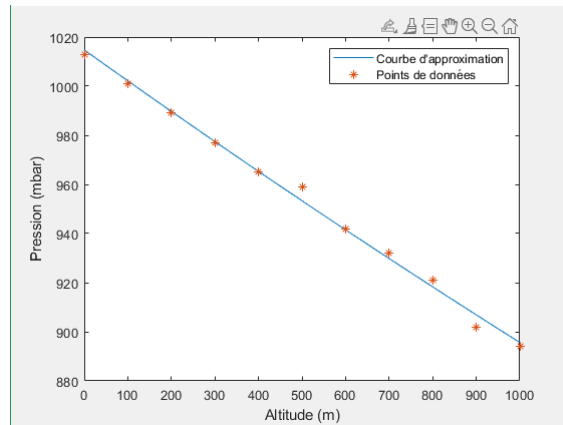


FIGURE 2.2 – Approximation exponentielle des données de pressions

Nous obtenons une exponentielle décroissante qui suit nos points établis.

## 2.3 Comparaison

Grâce à cette approximation, nous pouvons extrapoler nos pressions et ainsi déterminer par exemple la pression d'une atmosphère stable à 2000m d'altitude.




|   |   |          |
|---|---|----------|
| $p_{1500} = a \cdot \exp(k \cdot 1500) ;$ |   |          |
| $p_{2000} = a \cdot \exp(k \cdot 2000) ;$ |   |          |
| $p_{2500} = a \cdot \exp(k \cdot 2500) ;$ |   |          |
|   |  | p1500    |
|   |  | p2000    |
|   |  | p2500    |
|   |   | 841.4369 |
|   |   | 790.5099 |
|   |   | 742.6652 |

FIGURE 2.3 – Extrapolation sur trois points

Nous devons obtenir 843mbar pour 1500m, 795mbar pour 2000m et 747mbar pour 2500m.

Calculons l'erreur relative pour ces points.

$$\frac{ValeurThorique - Valeurpratique}{ValeurThorique}$$

$$Pour 1500m : \frac{843mbar - 841.4369mbar}{843mbar} = 1,854\%$$

$$Pour 2000m : \frac{795mbar - 790.5099mbar}{795mbar} = 5.648\%$$

$$Pour 2500m : \frac{747mbar - 742.6652mbar}{747mbar} = 5.803\%$$

Notre extrapolation est très cohérente. L'approximation exponentielle était un bon choix.