MT26 A 16

Projet : Interpolation

Rapport final

Robin Trioux, William Veal Phan

Introduction

L'interpolation d'une série de points est le processus visant à remplace un nombre fini de points, donc une représentation discrète d'un phénomène sur un segment donné, par une représentation continue, une fonction d'interpolation, sur ce même segment.

Il existe de nombreuses méthodes d'interpolation, utilisées dans de nombreux domaines scientifiques : en physique, avec de nombreuses applications au traitement de signaux, en statistique, mais aussi, et surtout, en informatique. L'interpolation permet la résolution approchée d'équations, la compression d'information ou encore le lissage de police d'écriture.

Dans le cadre de notre projet, nous nous proposons de suivre les instructions d'un sujet du semestre d'automne 2012. Nous étudierons donc l'interpolation de Lagrange, l'interpolation de l'Hermite, la méthode des moindres carrés polynomiaux et, enfin, l'interpolation trigonométrique.

Ces méthodes d'interpolation seront dans un premier temps implémentées sous forme de fonctions Maxima. Nous considèrerons que l'interface de wxMaxima, couplée au présent rapport et à une fonction d'aide, permettront d'utiliser de façon suffisante nos fonctions.

Table des matières

1. Méthodes d'interpolation implémentées

1.1. Interpolation de Lagrange

1.1.1. Principe général

L'interpolation selon Lagrange se propose d'interpoler *n+1* points (x_0,y_0), (x_0,y_0), (x_1,y_1), (x_2,y_2), (x_3,y_3) … (x_n,y_n) avec les x_i distincts deux à deux. On cherchera alors à construire un polynôme unique de degré au plus n ;

$$L(X) = \sum_{j=0}^{n} y_j . L_i(X)$$

Avec,

$$\forall i \in [\![0;n]\!]; \deg L_i \ = n \ et \ \ L_i(x_i) = \begin{cases} 1 \ si \ j = i \\ 0 \ sinon \end{cases}$$

Les polynômes de Lagrange sont exprimés :

$$\forall i \in [\![0;n]\!], L_i(\mathrm{X}) = \prod_{j=0; j \neq i}^{n} \frac{X - x_j}{x_i - x_j}$$

L'unicité de ce polynôme se prouve en exploitant les propriétés vectorielles des polynômes.

1.1.2. Algorithme

L'interpolation de Lagrange a été implémentée par deux fonctions, Lagrange et PedagoLagrange. Les deux fonctions ont un fonctionnement similaire, la seule différence étant que PedagoLagrange inclue l'affichage de graphique montrant la construction du polyn6ome d'interpolation.

```
Arguments :
xi : liste des abscisses, [x1,…,xn]
yi : liste des ordonnées, [y1,…,yn]
x : la variable muette
Résultat :
F_L(x) : une fonction de x, le polynôme d'interpolation
```

```
Lagrange(x : variable, xi : liste[réels],yi : liste[réels]) : fonction de
x
DEBUT
      n <- taille(xi)-1
      L(x) : fonction de x
      L(x) <- 0
      pour i de 1 a n+1 faire
            Li(x) fonction de x
            Li(x) <- 1
            pour j de 1 a n+1 faire
                  si (non(i = j)) alors
                        Li(x) <- Li(x) * (x-xi[j])/(xi[i]-xi[j])
                  fin si

            fait
            L(x) <- yi[i] * Li(x) + L(x)
      fait
      Lagrange <- L(x)
FIN
```

1.2. Interpolation de l'Hermite

1.2.1. Principe général

1.2.2. Algorithme

1.3. Méthode des moindres carrés

1.3.1. Principe général

La méthode des moindres carrés polynomiaux consiste cherche à ajuster une série de points par le polynôme du degré choisi minimisant la somme l'erreur quadratique.

Soit n un entier positif, et $\{(x_i, y_i)\ pour\ i \in [\![1; n]\!]\}$ une série de n points que nous souhaitons ajuster par un polynôme P de degré d :

$$\mathrm{P(X)} = \sum_{i=0}^{d} a_i X^i$$

On se fixe comme condition que P doit minimiser la somme du carré des erreurs, ie.

$$A = (a_0, \dots, a_d)\ minimise\ \sum_{i=0}^{n} (y_i - P(x_i))^2$$

On notera la solution

$$\hat{A} = \arg \min_{\hat{A} \in \mathbb{R}^{d+1}} \sum_{i=1}^{n} (y_i - P(x_i))^2$$

On détermine Â en posant :

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \text{ et } T = \begin{pmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^d \\ 1 & x_2 & {x_2}^2 & \cdots & {x_2}^d \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & {x_n}^2 & \cdots & {x_n}^d \end{pmatrix}$$

Et on peut montrer (la démonstration est possiblement au programme des UV MT25 et MT27) que

$$\hat{A} = (T^T T)^{-1} T^T Y$$

1.3.2. Algorithme

```
Arguments :
Lx : liste des abscisses des n points, [x1,x2,…,xn]
Ly : liste des ordonnées des n points, [y1,y2,…,yn]
d : le degré souhaité du polynôme

Résultat :
P(x) : polynôme d’interpolation minimisant l’erreur quadratique.

Fonction moindresCarres(Lx : liste[réels],Ly : liste[réels],d : entier
positif) : fonction de x
DEBUT
      n <- taille(Lx)
      T : matrice n x d+1
      Y : matrice n x 1
      P(x) : fonction de x
      P(x) <- 0
      pour k de 1 a n faire
            Y[k][1] <- Ly[k]
            pour j de 1 a d+1 faire
                  T[k][j] <- Lx[k]^(j-1)
            fait
      fait
      A : matrice 1 x d+1
```

```
    A <- inverse(transpose(T).T).transpose(T).Y
    pour k de 0 a d faire
        P(x) <- P(x) + A[k] * x^k
    fait
    moindresCarres <- P(x)
FIN
```

1.4. Interpolation trigonométrique

1.4.1. Principe général

L'interpolation trigonométrique consiste à interpoler une série de n points $((x_1, x_1), \dots, (x_n, x_n))$, en supposant la fonction f sous-jacente comme étant T-périodique, par un polynôme trigonométrique de degré au plus $d = \lfloor n/2 \rfloor$.

$$\forall x \in \mathbb{R}, P(x) = \sum_{k=-n}^{k=n} c_k e^{ik\frac{2\pi}{T}x} \textit{ avec } (c_n)_{n\in\mathbb{N}} \textit{ une suite de complexes ou de réels}$$

P peut aussi s'exprimer sous forme réelle

$$\forall x \in \mathbb{R}, P(x) = a_0 + \sum_{k\in[\![1,d]\!]} (a_k \cos\left(k\frac{2\pi}{T}x\right) + b_k \sin\left(k\frac{2\pi}{T}x\right))$$

Avec,

$$a_0 = \frac{1}{T}\int_0^T f(x)dx$$

Et pour k > 0,

$$a_k = \frac{2}{T}\int_0^T f(x)\cos\left(k\frac{2\pi}{T}x\right)dx \quad et\ b_k = \frac{2}{T}\int_0^T f(x)\sin\left(k\frac{2\pi}{T}x\right)dx$$

Les fonctions que nous allons approximer n'étant probablement pas périodique, nous allons poser que T, la période, est égale à notre intervalle d'interpolation, et, f étant inconnue dans le cas d'une interpolation, nous calculer numériquement les intégrales par la méthode des trapèzes.

Nous obtenons alors :

$$a_0 = \frac{1}{T}\sum_{j=1}^{n-1}\frac{1}{2}(y_{j+1} + y_j)(x_{j+1} - x_j)$$

Et pour k > 0,

$$a_k = \frac{1}{T}\sum_{j=1}^{n-1}(y_{j+1} + y_j)(x_{j+1} - x_j)\cos\left(k\frac{2\pi}{T}x\right)$$

$$et\ \ b_k = \frac{1}{T}\sum_{j=1}^{n-1}(y_{j+1} + y_j)(x_{j+1} - x_j)\sin\left(k\frac{2\pi}{T}x\right)$$

1.4.2. Algorithme

```
Arguments :
Lx : liste des abscisses, x1,…,xn
Ly : liste des ordonnées, y1,…,yn
Resultat :
f(x) : le polynôme trigonométrique ajustant la serie de points

fonction interpFourier(Lx : liste[réels], Ly : liste[réels]) : fonction de
x
DEBUT
      n : entier <- taille(Lx)
      T : réel <- Lx[n] -  Lx[1]
      w : réel <- 2*pi/T
      ak : réel
      bk : réel

      f(x) : fonction de x <- somme((Ly[j]+Ly[j+1])*(Lx[j+1]-
                                            Lx[j])/2,j,1,n-1)) / T

      pour k de 1 a valeurEntiere(n/2) faire
            ak <- somme((Ly[j] + Ly[j+1])*(Lx[j+1]-
                                   Lx[j])*cos(w*k*Lx[j]),j,1,n-1)) / T
            bk <- somme((Ly[j]+Ly[j+1])*(Lx[j+1]-
                                   Lx[j])*sin(w*k*Lx[j]),j,1,n-1)) / T
            f(x) <- f(x) + ak * cos(w*k*x) + bk * sin(w*k*x)
      fait

      interpFourier <- f(x)
FIN
```

À noter que, par soucis de performance et de lisibilité, nous avons utilisé des valeurs approchées des coefficients.

2. Étude et comparaison des méthodes d'interpolation

2.1. Listes de points

Afin de quantifier la qualité des différentes interpolations, nous avons besoin de pouvoir les tester sur des séries de points. Le sujet propose d'utiliser des distributions régulières, c'est-à-dire une série de points d'abscisses uniformément espacés, et des distributions de Tchebychev, ie.

Soit a et b deux réels ; a < b, n un entier et $k \in [\![0, n]\!]$;

$$\mathrm{Dist}_{\text{Régulière}} = \{x_k = a + \frac{k(b-a)}{n}\}$$

Et

$$\mathrm{Dist}_{\text{Tchebychev}} = \{x_k = \frac{a+b}{2} - \frac{b-a}{2} . \cos((k + \frac{1}{2})\pi \frac{1}{n+1}) \}$$

Ces distributions sont obtenues avec les fonctions

```
reglist : (a,b,n) -> x et tchebylist : (a,b,n) -> x
```

Avec a et b les bornes inférieures et supérieures, n l'indice du dernier point et x une liste d'abscisses.

Les ordonnées sont ensuite obtenues en appliquant la fonction à interpoler à la liste. Sous maxima, en posant u la fonction à interpoler et y la liste des ordonnées, on écrirait :

```
y : u(x) ;
```

On choisira une fonction u non polynomiale ou trigonométrique.

2.2. Définition du critère d'erreur

Afin de comparer la qualité de nos interpolations, nous avons besoin d'introduire un critère d'erreur général pouvant s'appliquer à toutes nos fonctions d'interpolation.

Soient a et b les bornes de notre interpolation, f la fonction que nous interpolons, et g la fonction d'interpolation. L'erreur quadratique est alors :

$$err(f,g) = \int_a^b (f(x) - g(x))^2 dx$$

Nous calculons cette erreur avec la fonction définie par :

```
Arguments :
f : fonction a interpoler
g : fonction d'interpolation
sous maxima, on donne les expressions de f(x) et g(x)
a,b : les bornes de l'intégration, a < b
Resultat :
err : un réel, l'erreur quadratique entre f et g

err(f : fonction, g : fonction, a : réel, b : réel) : réel
DEBUT
      err <- integrale((f(x)-g(x))²,x,a,b)
FIN
```

2.3. Comparaison Lagrange – l'Hermite

2.4. Fiabilité des moindres carrés polynomiaux

Conclusion