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 selon 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 considérerons que l'interface de wxMaxima, couplée au présent rapport et à une fonction d'aide, helpInterpol(), permettront d'utiliser de façon satisfaisante nos fonctions.

Table des matières

Introduction 2

Table des matières 3

1 Méthodes d'interpolation implémentées 3

1.1 Interpolation de Lagrange 4

1.1.1 Principe général 4

1.1.1 Algorithme 4

1.2 Interpolation selon Hermite 5

1.2.1 Principe général 5

1.1.2 Algorithme 5

1.3 Méthode des moindres carrés 6

1.3.1 Principe général 6

1.1.3 Algorithme 7

1.2 Interpolation trigonométrique 8

1.2.1 Principe général 8

1.3.2 Algorithme 9

2 Étude et comparaison des méthodes d'interpolation 9

2.1 Listes de points 10

Influence de la distribution 10

2.2 Définition du critère d'erreur 13

2.3 Comparaison Lagrange – Hermite 14

2.4 Comparaison moindres carrés polynomiaux – polynôme trigonométrique 18

2 Conclusion 21

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 ;

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

Avec,

$$\forall i,j \in (0;n); deg L_i = n \quad et \quad L_i(x_j)=\begin{pmatrix} 1 & si \ j=i \\ 0 & sinon \end{pmatrix}$$

Les polynômes de Lagrange sont exprimés :

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

L'unicité de ce polynôme se prouve en exploitant les propriétés les 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 polynôme d'interpolation étape par étape.

```
Arguments :
xi : liste des abscisses, [x0,…,xn]
yi : liste des ordonnées, [y0,…,yn]

Résultat :
F_L(x) : le polynôme de Lagrange
Lagrange(xi : liste[réels],yi : liste[réels]) : calcule F_L(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 selon Hermite

1.2.1 Principe général

La méthode d'interpolation d'interpolation décrite par Charles Hermite reprend les idées de Lagrange mais contraint d'avantage l'allure de l'interpolation de fonction ***f*** dérivable sur l'intervalle étudié. En effet, si la méthode de Lagrange impose que pour un échantillon $(\mathbf{x_i},\mathbf{y_i})$ considéré, $\mathbf{L(x_i)= y_i}$, la méthode d'Hermite impose en plus au polynôme de d'Hermite **H** une égalité sur les dérivées $\mathbf{H'(x_i)=f\,'(x_i)}$**.** Cela permet d'éviter dans certain cas les phénomène de Runge. On peut montrer que dans certains cas augmenter le nombre de point d'interpolation dégrade la précision de l'interpolation.

1.2.2 Algorithme

```
Arguments :
xi:Liste des abscisses des n+1 points.
yi:Liste des ordonnés des n+1 points.
Résultat :

L'expression du polynôme d'Hermite.
Hermite(xi : liste[réels],yi : liste[réels]):

DEBUT
    n ← taille(xi)
    H ← 0
    pour i de 1 à n en 1 faire
        Li ← 1
        pour j de 1 à n en 1 faire
            si i ≠ j
                alors
                (Li:Li*((x-xi[j])/(xi[i]-xi[j])))
            fin si
        qi ← Li²
        F_qi(x) ← qi
        F_Li(x) ← Li
        F_dLi(x)← dLi/dx

        Hi ← qi*(1-2*F_dLi(xi[i])*(x-xi[i])),
        Ki ← qi*(x-xi[i]),
        si i<n et i≠1
            alors
            DeltaY ← ((yi[i+1]-yi[i])/(xi[i+1]-xi[i])+
                        (yi[i]-yi[i-1])/(xi[i]-xi[i-1]))/2
            H ← H+(yi[i]*Hi+DeltaY*Ki)
```

```
    F_H(x) ← H
    Hermite(xi,yi) ←  F_H(x)
FIN
```

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 du carré des erreurs.

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 :

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

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

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

On détermine Â en posant :

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

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 $((x_1, x_1), \ldots, (x_n, x_n))$, en supposant la fonction f sous-jacente comme étant T-périodique, par un polynôme trigonométrique de degré au plus .

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

P peut aussi s'exprimer sous forme réelle

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

Avec,

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

Et pour k > 0,

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

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,

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

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

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

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

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.

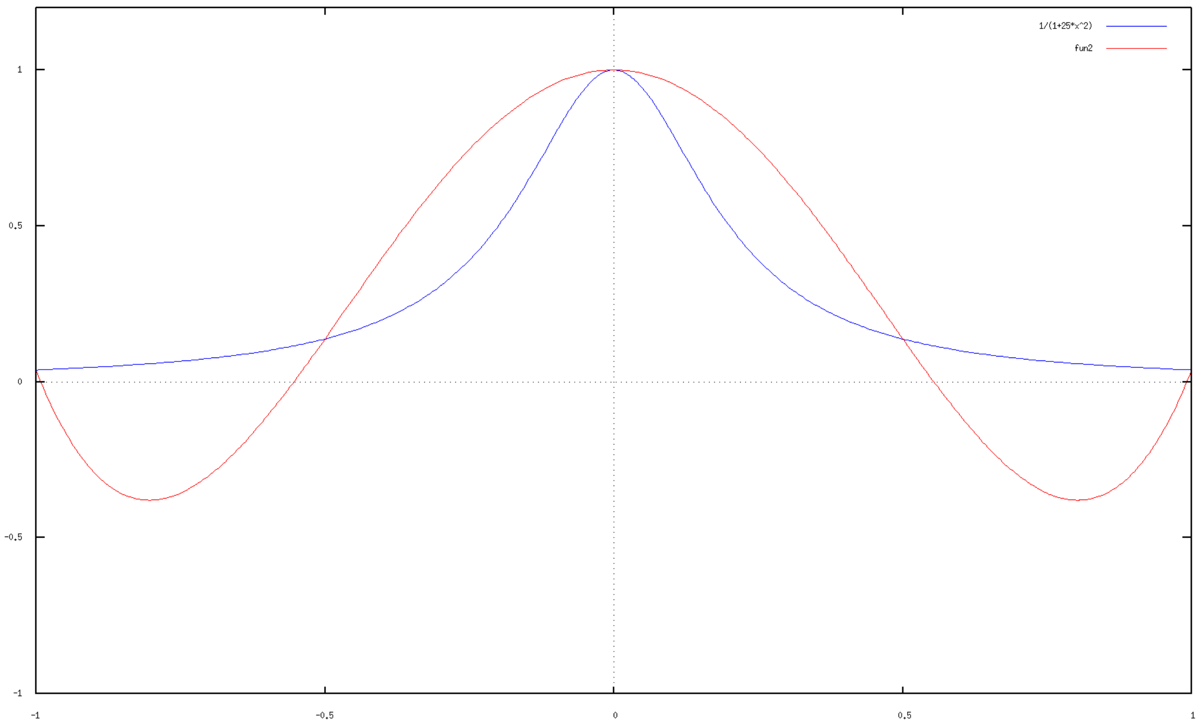
2.1.1 Influence de la distribution

Le choix de la distribution de point peut être déterminante quant à la qualité de l'interpolation. Nous allons l'illustrer sur l'interpolation de la fonction $f(x)=\frac{1}{1+25.x^2}$ qui est sujette aux phénomènes de Runge lors de l'interpolation Lagrangienne.

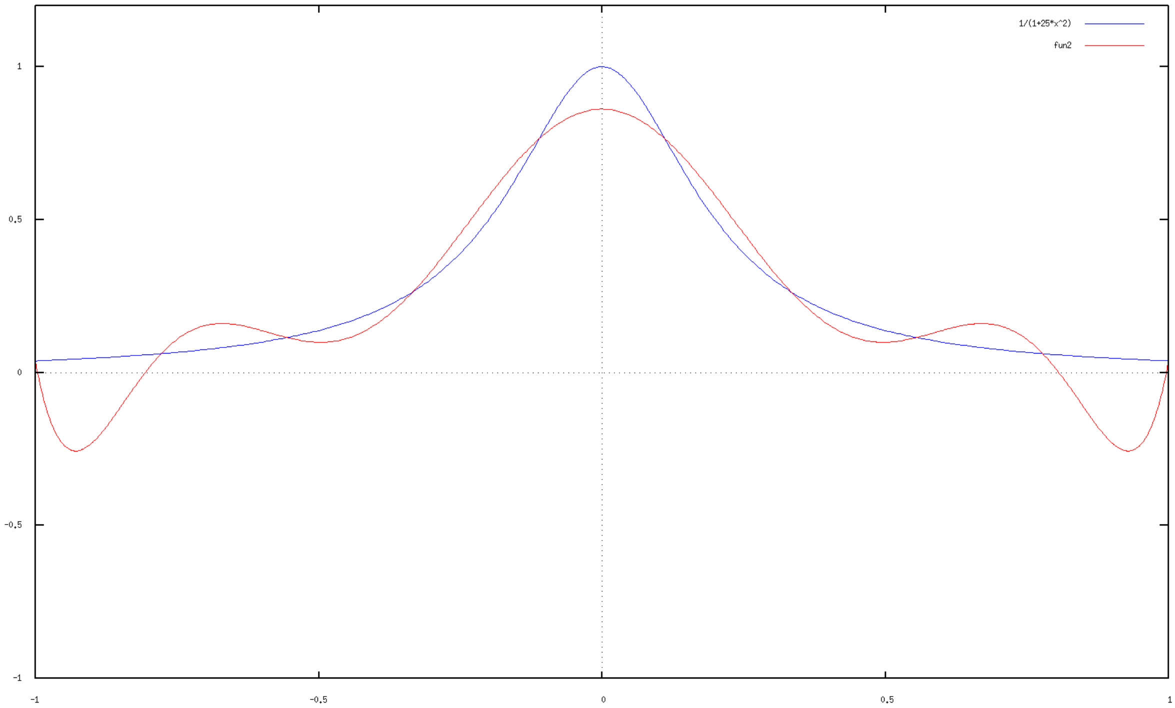
On représente en bleu le graph de la fontion f et rouge celui de L, l'interpolation par Lagrange sur [-1,1] de la fonction f.

nb : Nous avons affiché Hermite et Lagrange sur deux graphiques séparés car afficher demander le calcul et l'affichage faisait « crasher » l'ordinateur.

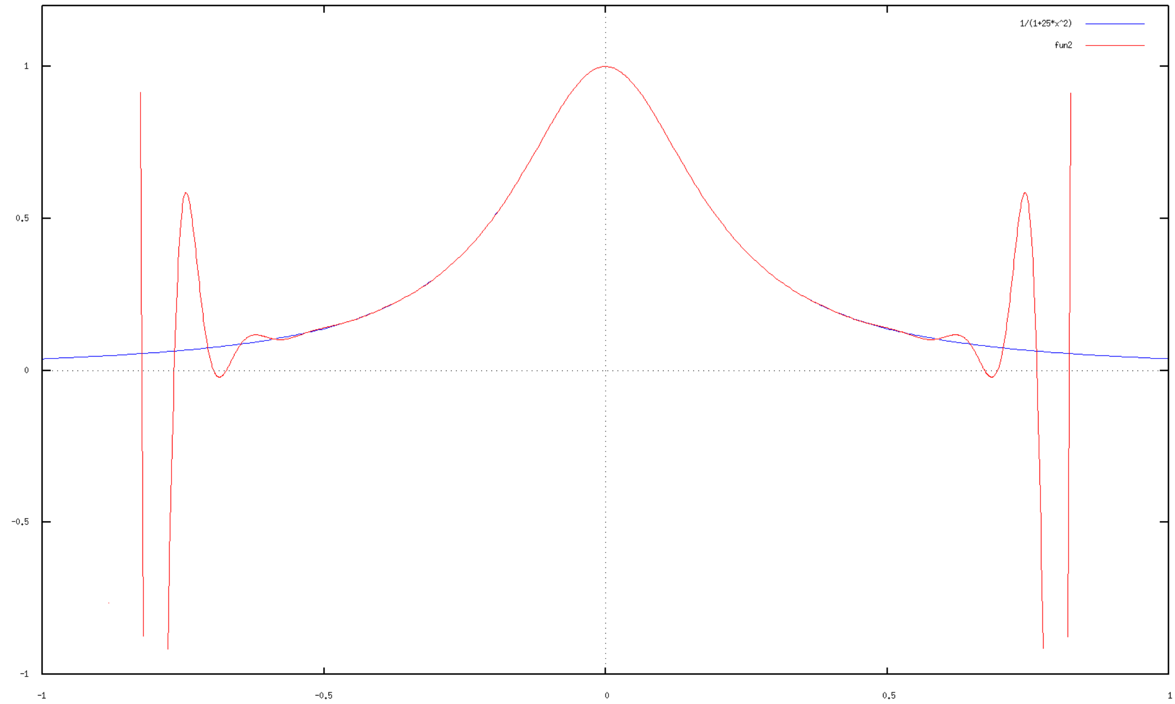
Pour **n=5**



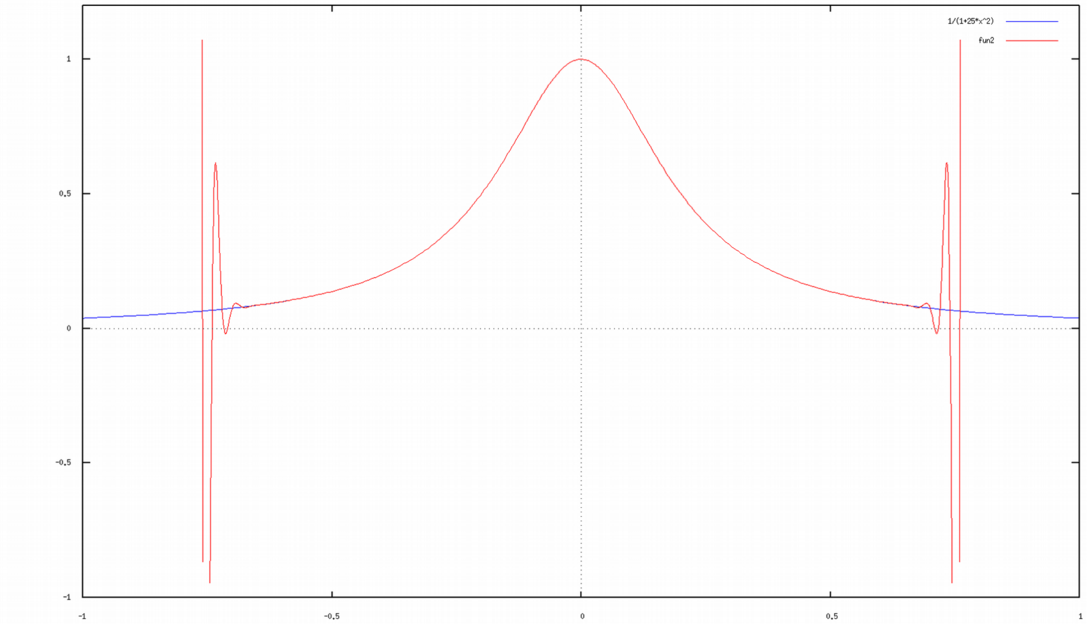
Pour **n=10**



Pour **n=35**

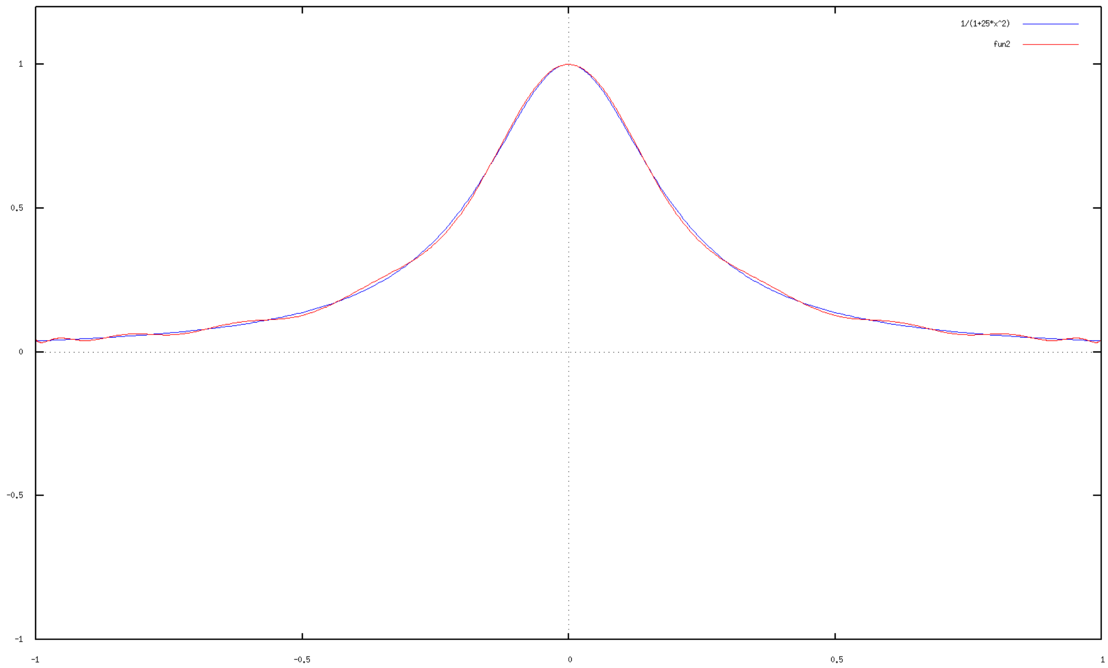


Pour **n=100**

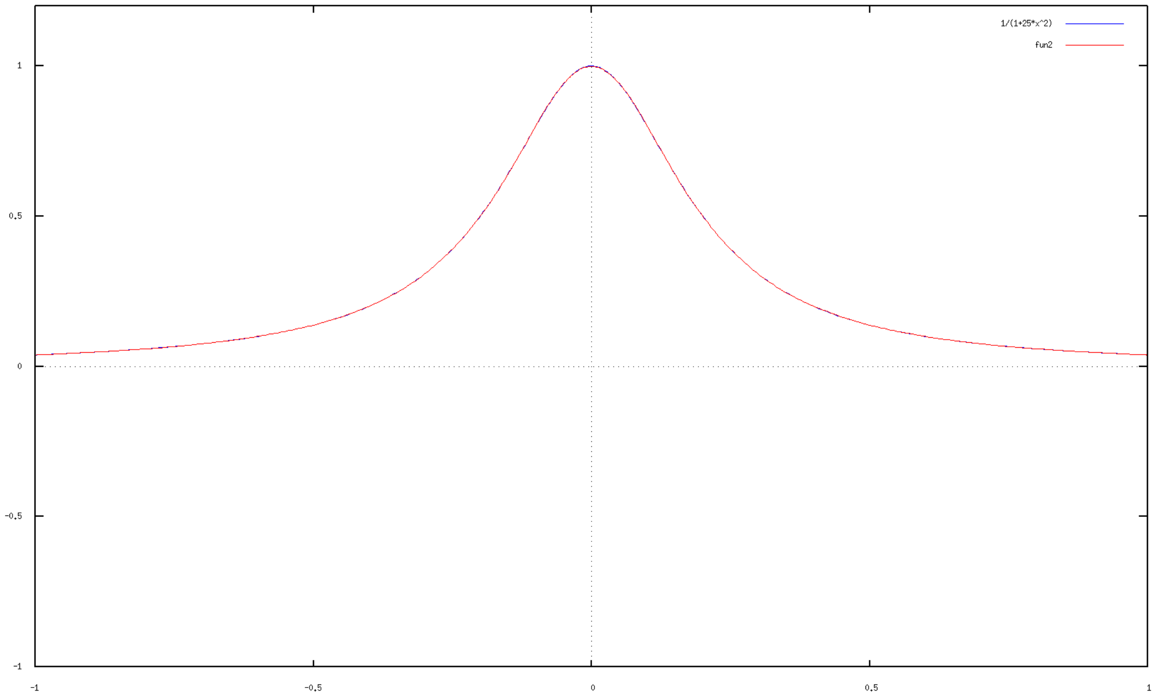


On voit que le nombre de point ne permet pas une très bonne interpolation. Seulement si l'on choisis une distribution dite de Tchebychev, nous obtenons de bien meilleurs résultats.

Pour **n=20**



Pour **n=35**



Les deux courbes se confondent, ce qui illustre l'importance du choix de la distribution de point.

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, ou erreur de norme $L^2(a,b)$, 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
a,b : les bornes de l'intégration, a < b
Resultat :
err : un réel, l'erreur quadratique entre f et g

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

Nous avons aussi choisi d'utiliser la somme des erreurs au carré (fonction errQuadDiscrete) dans les cas où les calculs pour la précédente fonction excèdent la capacité de calcul de Maxima.

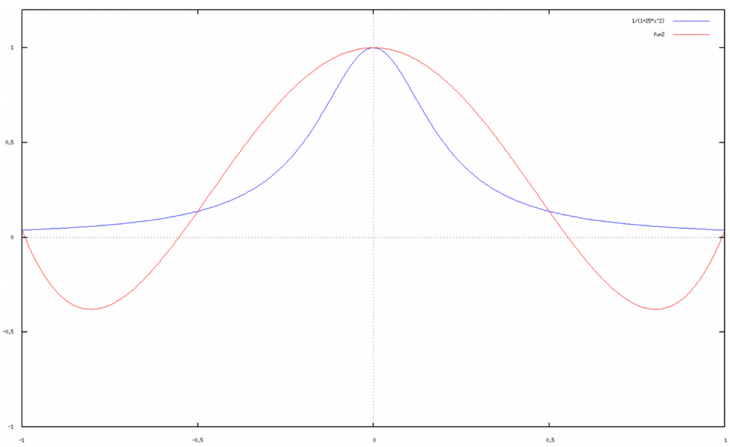
2.3 Comparaison Lagrange – Hermite

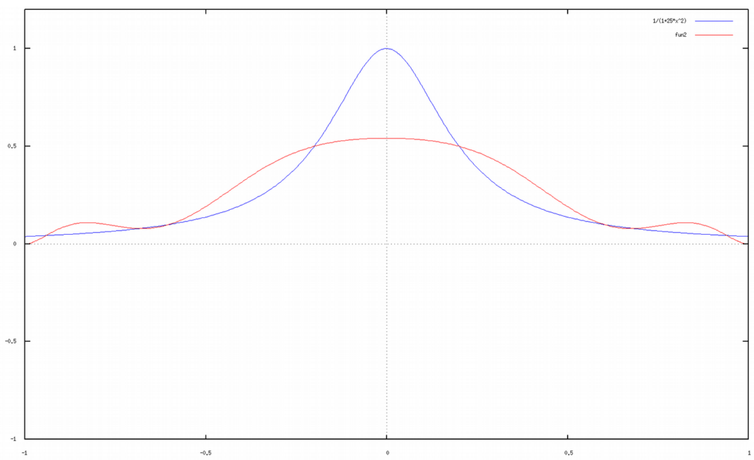
En théorie, l'interpolation d'Hermite permet une meilleure approximation de fonction dérivable que Lagrange car elle impose des contraintes sur les dérivées, ce qui double le nombre de polynôme à calculer et par conséquent l'inertie des calculs. En outre, le calcul de H nécessite la connaissance de **f '(x)**, donnée supposée inaccessible. Nous devons donc faire des approximations sur les **f'(xi).** Nous avons donc besoin d'un grand nombre de points pour affiner cette approximation. En particulier nous ne nous attendions pas à ce que l'interpolation d'Hermite échoue lors de l'interpolation de la fonction $f(x)=\frac{1}{1+25.x^2}$, fonction qui rappelons le est sujette aux phénomènes de Runge.

Pour la fonction **f, distribution régulière** :

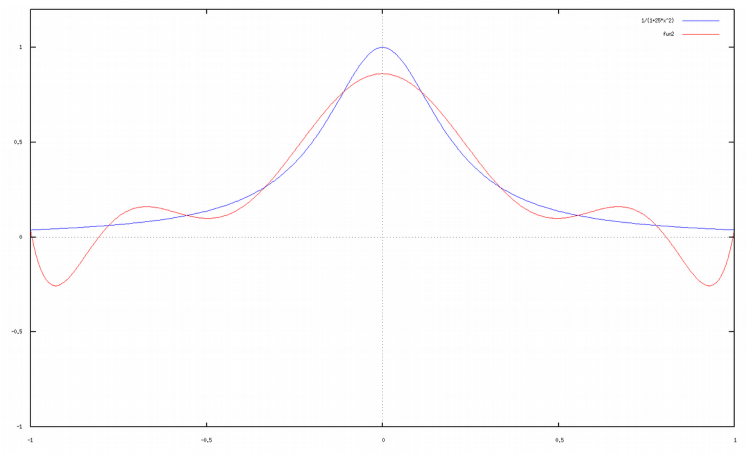
A gauche Lagrange et à droite Hermite :

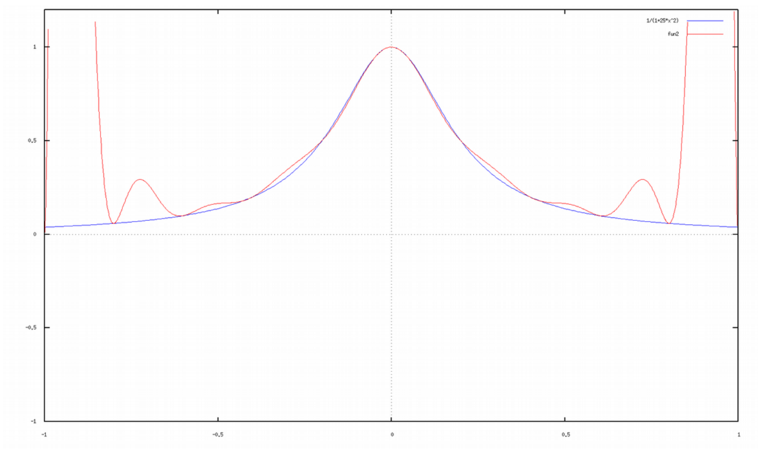
Pour n=5 :



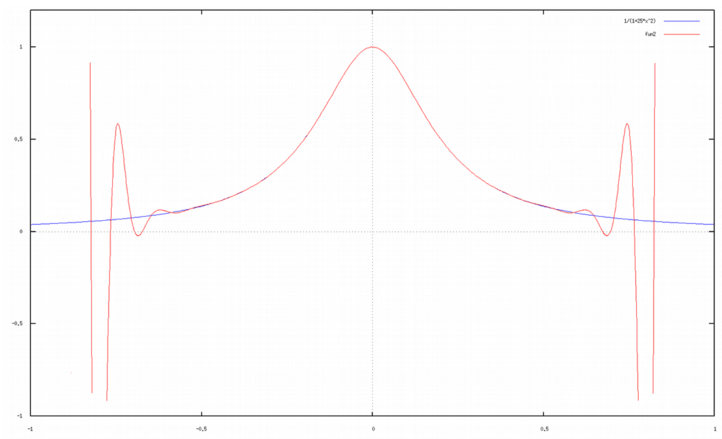


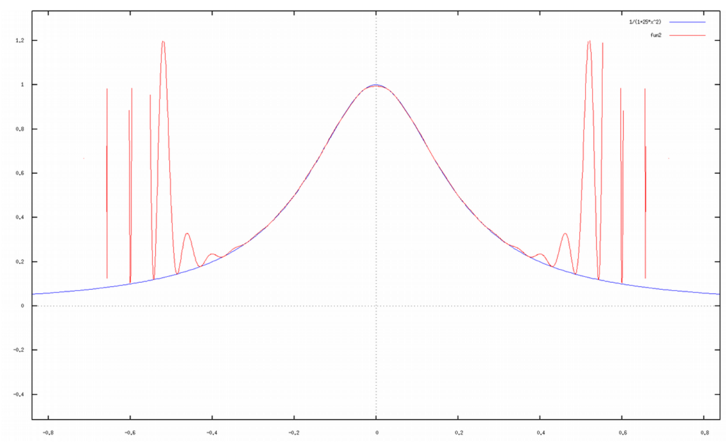
Pour n=10 :



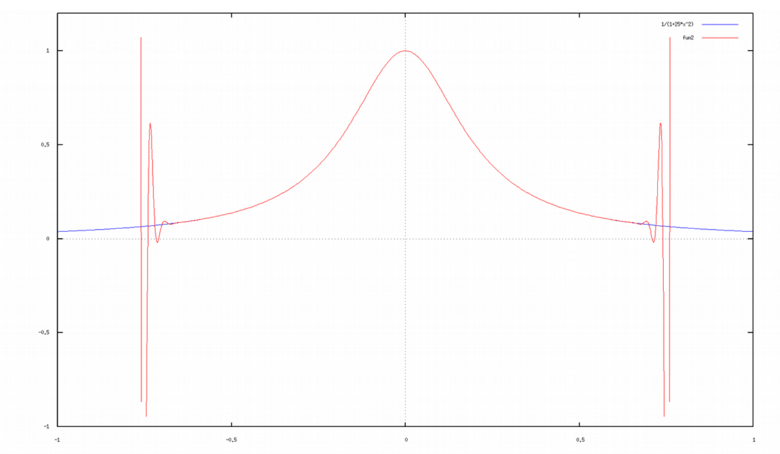


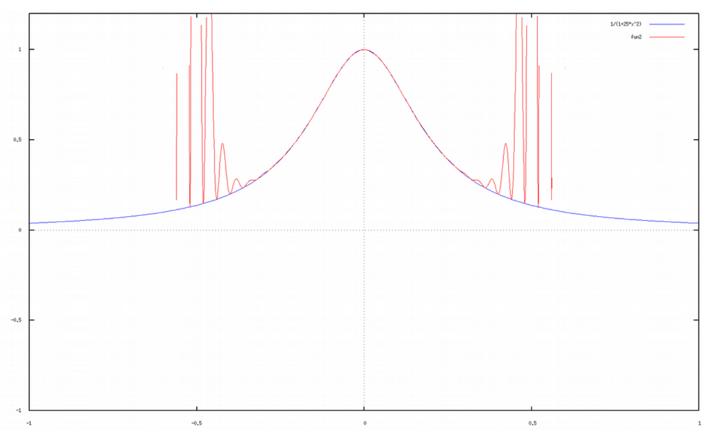
Pour **n=35**





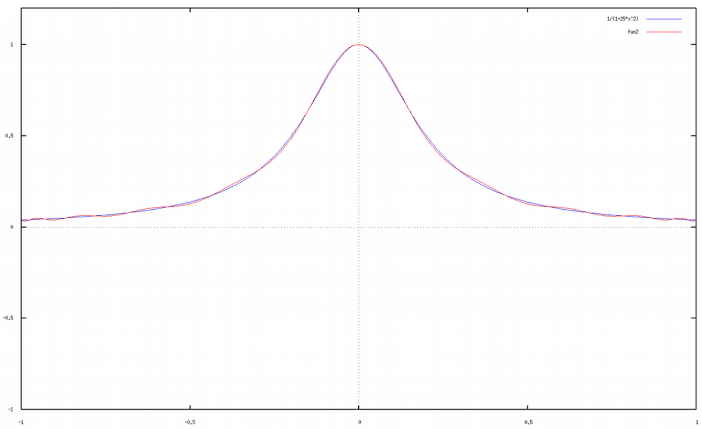
Pour n=100

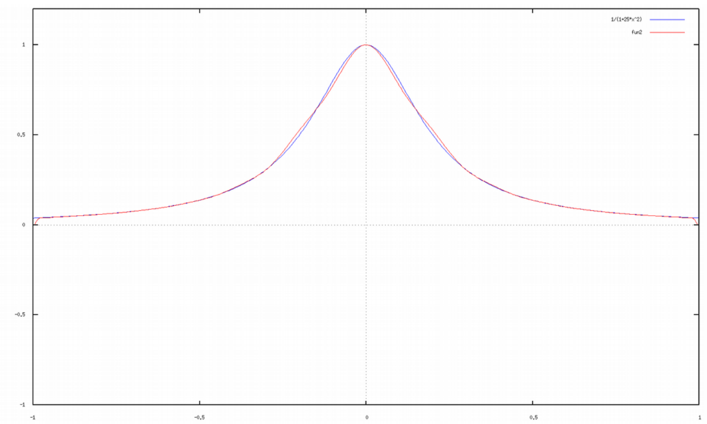




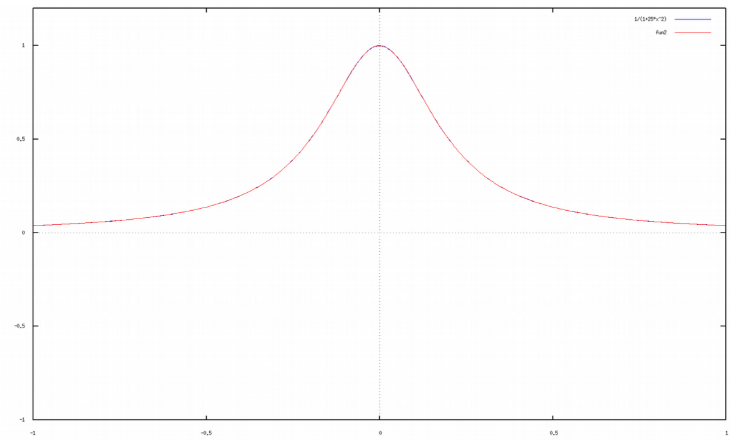
Il semble que sur la distribution régulière, pour de faible valeur de n soit de meilleure qualité pour Hermite mais cela tend à s'inverser au fur et à mesure que n croît.
Regardons rapidement si cela est valable également pour la **distribution de Tchebychev.**

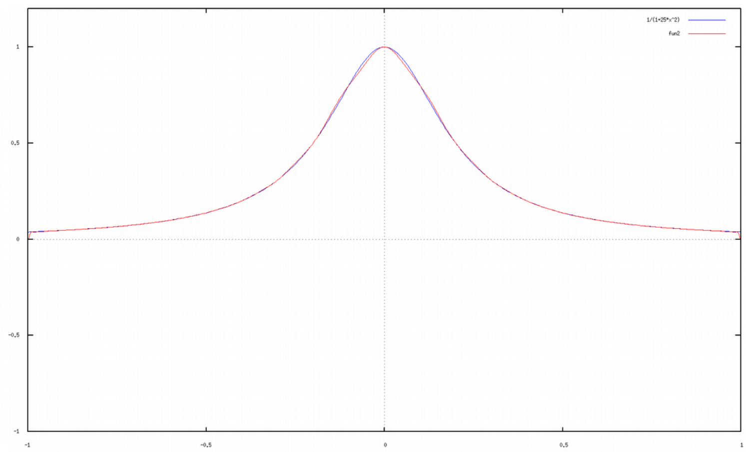
Pour n=20





Pour n=30



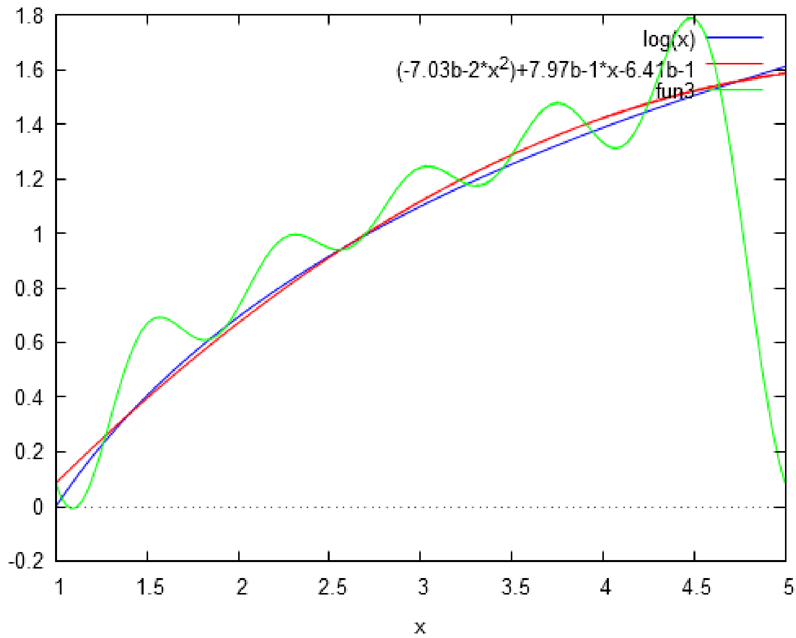


Les conclusions sont les mêmes pour la distribution de Tchebychev.

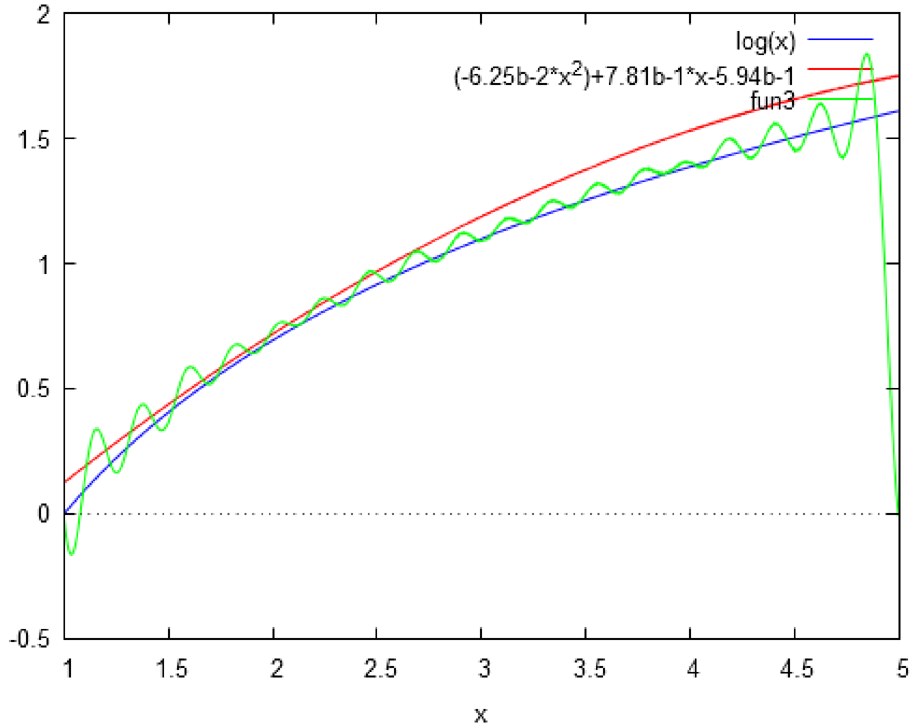
2.4 Comparaison moindres carrés polynomiaux – polynôme trigonométrique

Nous nous intéressons maintenant aux interpolations par les moindres carrés polynomiaux et l'interpolation par un polynôme trigonométrique, elle aussi une approximation par moindre carré.

Nous choisissons d'utiliser comme fonction à interpoler la fonction ln entre 1 et 5, et générons une distribution régulière avec la fonction *reglist*.



Nous avons ici en vert le polynôme trigonométrique F, en rouge le polynôme des moindres carrés P, et en bleu le graph de ln. Nous avons choisi P de degré 2. On remarque que le nombre de points semble insuffisant pour assurer une bonne précision de l'interpolation trigonométrique. Si nous augmentons le nombre de points à 35 :



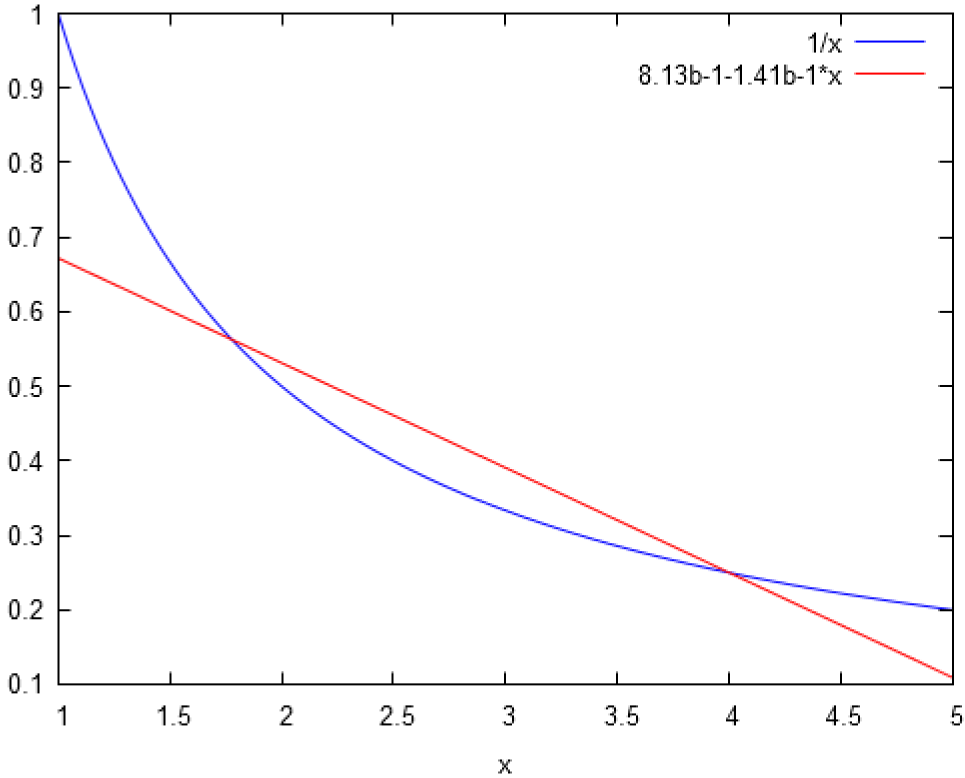
Nous observons alors que l'interpolation trigonométrique semble plus précise tandis que l'interpolation polynomiale semble perdre en fiabilité. Nous vérifions en calculant les erreurs quadratiques.

$err_{10}(\ln,P) = 0.004414236243206915$ et $err_{35}(\ln,P) = 0.04088067925470362$

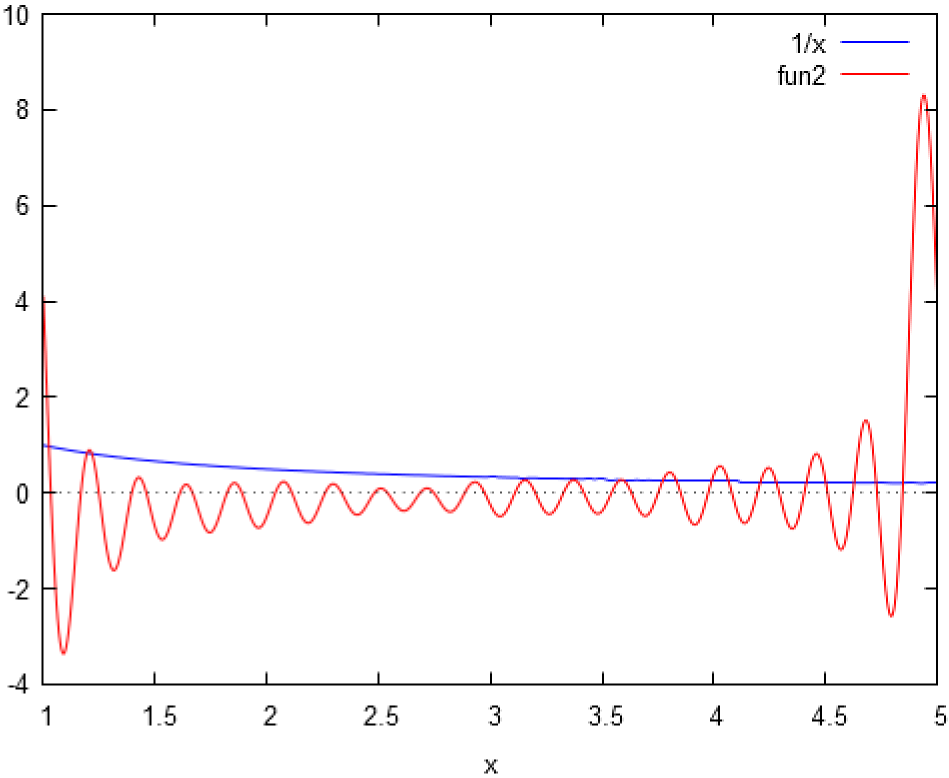
tandis que $err_{10}(\ln,F) = 2{,}76$, et $err_{35}(\ln,F) < err_{10}(\ln,F)$

Il apparaît donc que la précision et la fiabilité de l'interpolation trigonométrique augmente avec le nombre de points à interpoler. Au contraire, on peut voir, en testant pour d'autres degrés de P ou pour différents nombres de points, que la fiabilité de l'interpolation trigonométrique varie considérablement.

Nous cherchons ensuite à déterminer si ces interpolations de points du graph de ln nous permettent aussi d'obtenir une approximation de ln'. Nous testons donc pour la même distribution régulière de 35 points :



Nous pouvons ici observer que P' peut être vue comme une interpolation linéaire de la dérivée de ln. Au contraire :



F' ne permet pas du tout de conclure quoique ce soit sur la dérivée de ln.

3 Conclusion

Somme toute, aucune méthode d'interpolation n'est à privilégier absolument.

L''aspect de la distribution joue aussi un rôle important dans le choix de l'interpolante à utiliser. On retiendra cependant que, hors cas particulier, l'interpolation trigonométrique est à préférer dans le cas d'un signal périodique, et que, bien que plus contraignante, l'interpolation d'Hermite est plus précise que l'interpolation de Lagrange aux points d'interpolations.