LOGICIELS LIBRES ET SIMULATIONS NUMÉRIQUES
INTERPOLATION POLYNOMIALE - SAGEMATH/PYTHON

Ines Abdeljaoued Tej [1]

1 Méthode de Lagrange

Créez un fichier dans votre espace de travail. Tous vos résultats doivent être enregistrés dans ce fichier. Dans cette séance, le but est de tester l'interpolation polynomiale et de se familiariser avec le phénomène de Runge :

1.1 Prévision d'un temps de calcul lors d'un processus industriel

Il est assez fréquent en milieu industriel qu'à l'occasion de l'achat d'un gros matériel par un client, l'acheteur potentiel vienne vérifier sur site que l'étude théorique des performances attendues est attestée par l'expérience. La qualité de prévision doit être irreprochable. Dans le cadre d'un processus industriel mené en temps réel, l'ingénieur doit attendre le résultat d'un traitement informatique lourd exécuté par un algorithme nommé $Traitement(N)$ dont la durée $T(N)$ dépend de la taille N des données traitées. Lors de 3 essais, l'ingénieur a noté selon la valeur de n le temps de calcul $T(N)$ en milliers de secondes nécessaires [2]. Cette expérimentation a donné lieu au tableau suivant :

N	0	1	2	3
$T(N)$	0	1	4	9

1. Trouver le polynôme d'interpolation P de T en utilisant l'un des algorithmes vus en cours.
2. En déduire le temps $T(N)$ nécessaire au traitement de données de taille N
3. Représentez le polynôme P.
4. Trouvez la fonction Python permettant de répondre à ce problème.

1.2 Implémentation de l'interpolation de Lagrange

Notons xdata la liste contenant les valeurs expérimentale choisies pour N et fdata les résultats des mesures effectuées ($T(N)$). Corrigez le script suivant pour obtenir le polynôme d'interpolation `P` :

1. `inestej@gmail.com, http://bit.ly/1JEWZiK`
2. TP inspiré du livre de J. Bastien et J.-N. Martin intitulé *Introduction à l'analyse numérique - Application sous Matkab* aux éditions DUNOD, Année 2003. Livre disponible à la bibliothèque de l'ESSAI (code 1485).

```
xdata=[0,1,2,3]
fdata=[0,1,4,9]
n=len(xdata)-1
P=0
for i in range(n+1):
...       L=1
...       for j in range(n+1):
...           if i<>j:
...               L=L*(x-xdata[j])/(xdata[i]-xdata[j])
...       P=P+fdata[i]*L
expand(P)
```

La solution est égale au polynôme x^2 qui est de degré ≤ 3 passant par les points d'interpolation (xdata[i], fdata[i]) pour i de 0 à 3. Dans le cas du processus industriel, $T(N) = N^2$.

2 Erreur d'interpolation

Théorème : Soient f une fonction de classe C^{n+1} sur un intervalle $[a, b]$ et P le polynôme de Lagrange de f en les points $x_0 < x_1 < \ldots < x_n \in [a, b]$. Alors

$$f(x) - P(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_n)}{(n + 1)!}\, f^{(n+1)}(\zeta) \tag{1}$$

où $a \leq min(x, x_0) < \zeta < max(x, x_n) \leq b$.

Démonstration : La relation est triviale pour $x = x_i \ \forall i = 0..n$. Soit $x \neq x_i$, pour tout $i = 0..n$. On pose $k(x) = \frac{f(x)-P(x)}{(x-x_0)(x-x_1)\ldots(x-x_n)}$ et $w(t)$ un polynôme qui s'annule en x_0, x_1, …, x_n, et x définit par : $w(t) = f(t) - P(t) - (t - x_0)(t - x_1)\ldots(t - x_n)k(x)$. Il s'annule en $(n + 2)$ points. En appliquant le thórème de Rolle, on montre que $w^{n+1}(t)$ s'annule en au moins un point ζ :

$$w^{(n+1)}(\zeta) = f^{(n+1)}(\zeta) - (n + 1)!\, k(x) = 0 \qquad et \qquad k(x) = \frac{f^{(n+1)}(\zeta)}{(n + 1)!} \qquad .$$

Cette quantité ne tend pas nécessairement vers 0 car les dérivées $f^{(n+1)}$ de f peuvent grandir très vite lorsque n croît. Dans l'exemple suivant, nous présentons deux cas sur la convergence de l'interpolation de Lagrange.

2.1 Exemple

Soit $f(x) = \sin(x)$, $\mid f^k(x) \mid \leq 1$: l'interpolé P converge vers f quelque soit le nombre de points d'interpolation et leur emplacement. Par contre, pour $f(x) = \frac{1}{1+14x^2}$ sur $[-1, 1]$ et bien que f soit indéfiniment continûment dérivable sur $[-1, 1]$, les grandeurs

$$max_{-1\leq x\leq 1} \mid f^k(x) \mid \qquad , \qquad k = 1, 2, 3, \ldots$$

explosent très rapidement et ceci ne nous assure plus la convergence de l'interpolation.

Le choix des points x_i apporte une amélioration sensible de l'interpolation : Lorsque $a \leq x \leq b$, on choisit les abscisses d'interpolation

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i+1}{2(n+1)}\pi)$$

pour $i = 0..n$, on obtient :

$$\mid f(x) - P(x) \mid \leq \frac{(b-a)^{n+1}}{(n+1)!2^{2n+1}}\, max_{x\in[a,b]} \mid f^{(n+1)}(x) \mid \quad .$$

Les points $x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i+1}{2(n+1)}\pi)$ pour $i = 0..n$ sont calculés à partir des zéros des polynômes de Tchebychev ($T_n(x) =$ vérifiant $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$).

A faire sur Python :

1. Soit $f(x) = sin(x)$ et une subdivision de l'intervalle $[-1, 1]$ en $n = 5$ sous-intervalles equidistants. Posons $x_0 = -1$, $x_1 = -0.5$, $x_2 = 0$, $x_3 = 0.5$ et $x_4 = 1$.
   Implémenter l'algorithme de Lagrange se basant sur les polynômes d'interpolation de base. On prendra $f_i = f(x_i)$ pour $i = 0..n$.
   Dessiner dans un même graphique la fonction f, le polynôme P ainsi que les points (x_i, f_i). Reprendre les calculs pour $n = 13$.
2. Refaire le même travail avec la fonction $f(x) = \frac{1}{1+14x^2}$, puis avec $x_i = \cos(\frac{2i+1}{2n+2}\pi)$ pour $i = 0..n$ et $n = 5$ puis $n = 13$.

2.2 Phénomène de Runge

Le code Python est donné ci-dessous :

```
 : from pylab import arange
 : xdata=arange(-1.0,1.1,0.5)
 : f=sin
 : P=lagrange(f, xdata)
 : R=plot(f, -1,1, rgbcolor=(0,2,1))
 : Q=plot(P, -1,1, rgbcolor=(0,1,2))
 : n=len(xdata)-1
 : s=Graphics()
 : for i in range(n+1):
...           xi = xdata[i]
...           fi = f(xdata[i])
...           s = s + point((xi,fi), rgbcolor = (1,0,0))
 : R+Q+s
```

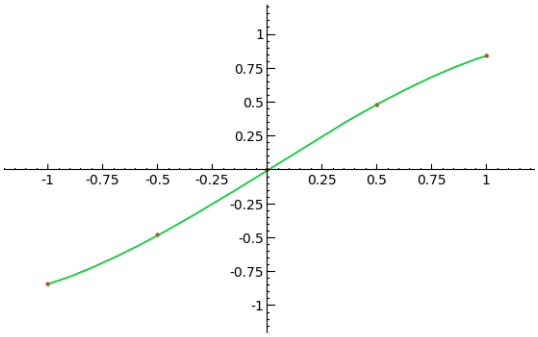

FIGURE 1 – *Interpolation pour* $n = 5$

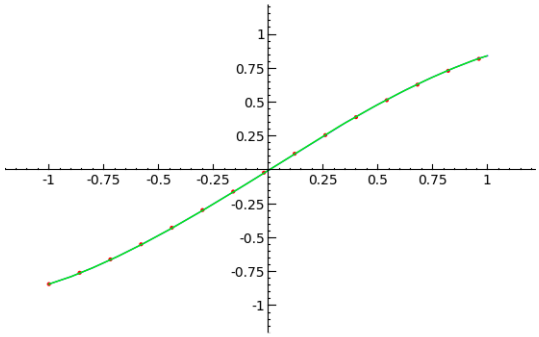

FIGURE 2 – *Interpolation pour* $n = 13$

Les graphiques suivants donnent la fonction *sin* et son polynoôme de Lagrange sur $n = 5$ puis sur $n = 13$ points :

2. Si on calcule le polynôme de Lagrange sur la fonction $f = \frac{1}{1+14x^2}$, nous obtenons :

```
 : from pylab import arange
 : xdata=arange(-1.0,1.1,0.5)
 : f=1/(1+14*x^2)
 : P=lagrange(f, xdata)
 : R=plot(f, -1,1, rgbcolor=(0,2,1))
 : Q=plot(P, -1,1, rgbcolor=(0,1,2))
 : n=len(xdata)-1
 : s=Graphics()
 : for i in range(n+1):
...           xi = xdata[i]
...           fi = f(xdata[i])
...           s = s + point((xi,fi), rgbcolor = (1,0,0))
...
 : R+Q+s
```

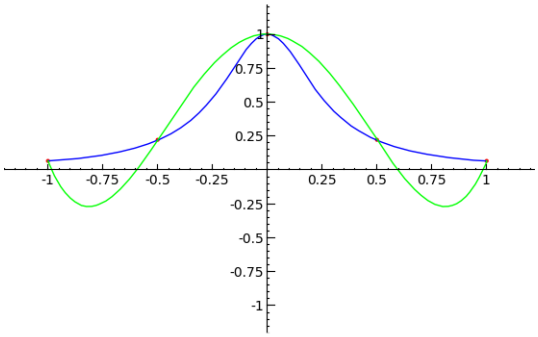

FIGURE 3 – *Interpolation pour* $n = 5$

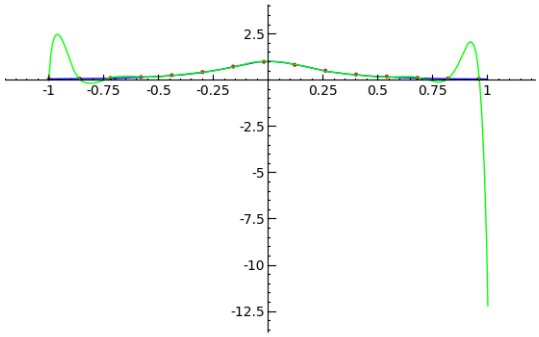

FIGURE 4 – *Interpolation pour* $n = 13$

Enfin, si on considère à la place des abscisses x_i équidistants, les zéros de Tchebychev, nous obtenons la figure 5 pour $n = 5$ points et ensuite la figure 6 pour $n = 13$:

```
 : a=-1; b=1; n=5;
 : ydata=[simplify((a+b)/2+(b-a)/2*cos((2*k+1)*3.14/(2*n+2))) for k in range(n+1)]
 : print(ydata)
[0.965960168538399, 0.707388269167200, 0.259459981914882, -0.257921542147459,
-0.706261644820005, -0.965546938710468]
 : f=1/(1+14*x^2)
 : P=lagrange(f, ydata)
 : A=plot(f, -1,1, rgbcolor=(0,2,1))
 : B=plot(P, -1,1, rgbcolor=(1,0,0))
 : A+B
```

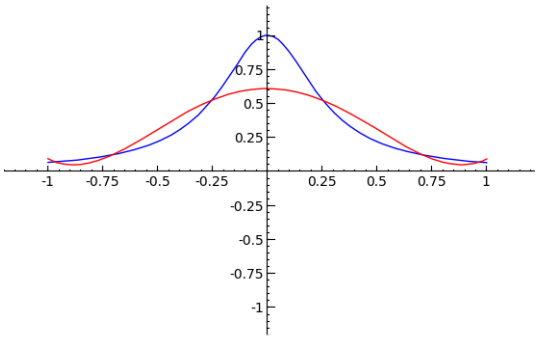

FIGURE 5 – *Interpolation sur les zéros de Tchebychev pour $n = 5$*

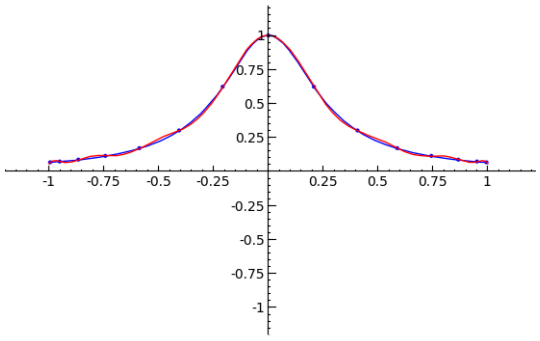

FIGURE 6 – *Interpolation sur les zéros de Tchebychev pour $n = 13$*

```
: a=-1; b=1; n=13;
 : ydata=[simplify((a+b)/2+(b-a)/2*cos((2*k+1)*3.14/(2*n+2))) for k in range(n+1)]
 : print(ydata)
[0.993718576879459, 0.943939675865892, 0.846875476196380, 0.707388269167200,
 0.532465465533183, 0.330869571228815, 0.112699242252689, -0.111116592961247,
-0.329366202474389, -0.531116686408463, -0.706261644820005, -0.846027443250886,
-0.943412715311215, -0.993539086045688]
 : f=1/(1+14*x^2)
 : P=lagrange(f, ydata)
 : A=plot(f, -1,1, rgbcolor=(0,2,1))
 : B=plot(P, -1,1, rgbcolor=(1,0,0))
 : A+B
```

3 Formule de Horner

Il est plus efficace d'écrire le polynôme de Lagrange dans la base de polynômes de Newton : $\left\{1, (x-x_0), (x-x_0)(x-x_1), \ldots, \prod_{i=0}^{i=n-1}(x-x_i)\right\}$. Implémentez la méthode de Newton ainsi que la formule de Horner pour spécialiser un polynôme.