InterpolationEtApproximation

Sébastien Joannès

September 18, 2015

Contents

1 Interpolation & Approximation **1**
1.1 Quand et pourquoi interpoler ou approximer ? 1
1.2 Utiliser des polynômes pour interpoler/approximer 2

2 Les polynômes de Lagrange au coeur de l'interpolation **2**
2.1 Formulation lagrangienne 2
2.2 Phénomène de Runge 5
2.3 Polynômes de Chebyshev et atténuation du phénomène de Runge 9
2.4 Formulation newtonienne et différences divisées 12
2.5 Formulation barycentrique 14
2.6 Formulation d'Hermite, prise en compte des dérivées 15

1 Interpolation & Approximation

D'après le Dictionnaire Historique de la langue française d'Alain Rey (Le Robert), le terme latin interpolare signifie "refaire, donner une nouvelle forme". Il pourrait être issu des termes inter- ("entre") et polire (polir, rendre uni). L'action d'**interpoler** n'apparait en mathématiques qu'en 1812.

L'**interpolation** désigne la construction d'une courbe à partir de la donnée d'un nombre fini de points, ou d'une fonction à partir de la donnée d'un nombre fini de valeurs. La solution du problème d'interpolation passe nécessairement et au minimum par les points prescrits et peut nécessiter de respecter des contraintes supplémentaires.
L'**interpolation** doit être distinguée de l'**approximation**, qui consiste alors à trouver la fonction la plus proche possible d'une série de données. Dans le cas de l'approximation, il n'est en général plus imposé de passer exactement par les points initiaux.
Cependant, une confusion est souvent possible et l'on parle également d'**approximation** de la fonction pour désigner les valeurs (estimées) issues d'une **interpolation** ...

1.1 Quand et pourquoi interpoler ou approximer ?

Certaines fonctions, inconnues explicitement, sont simplement évaluables par des calculs coûteux. Plutôt que de recourir à une évaluation à la demande, une base de données (abaques) est

généralement préétablie. La connaisance de la fonction en un point quelconque passe alors par une **interpolation**.

La liste ci-desous regroupe quelques exemples pouvant nécessiter (ou ayant nécessité par le passé) le recours aux techniques d'**interpolation**:

- Tables de logarithmes, tables trigonométriques, ...
- Abaques aéronautiques, ...
- Résolution d'une équation aux dérivées partielles par la méthodes des **éléments finis**
- Imagerie (reconstruction)

Le recours à l'**approximation** est nécessaire lorsque les données forment par exemple un nuage de points (statistiques). Une tendance est alors recherchée.

1.2 Utiliser des polynômes pour interpoler/approximer

Pour représenter une fonction f par une fonction simple et facile à évaluer, il faut se restreindre à une famille de fonctions dont la manipulation est aisée. Les choix les plus pertinents se basent sur:

- Les polynômes
- Les fonctions trigonométriques
- Les fonctions logarithmiques ou exponentielles

Le **théorème d'approximation de Weierstrass** (1885) justifie en particulier le recours aux polynômes pour interpoler une fonction f suffisamment régulière.
Toute fonction f continue sur un segment $[a, b]$ est limite uniforme de fonctions polynomiales sur ce segment $[a, b]$. En d'autres termes, pour tout $\varepsilon > 0$, il existe une fonction polynomiale p à coefficients réels telle que:

$$|f(x)–p(x)| < \varepsilon \quad \forall x \in [a, b]$$

Il existe également un version **trigonométrique** de ce théorème (issu de la théorie des séries de Fourier). Pour toute fonction f continue et périodique, il existe une suite de polynômes trigonométriques qui converge uniformément vers f.

2 Les polynômes de Lagrange au coeur de l'interpolation

2.1 Formulation lagrangienne

Les polynômes de **Lagrange** ou plus précisement polynômes de **Waring(1779)-Euler(1783)-Lagrange(1795)** offrent un cadre idéal pour l'interpolation.
Soit $x_0, x_1, \ldots, x_n$, $n+1$ réels distincs. Il existe $n+1$ polynômes L_i pour $i = 0$ à n définis par:

$$L_i(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)} \tag{1}$$

Ou sous une forme contractée:

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \tag{2}$$

Les propriétés suivantes peuvent notamment être démontrées:

- L_i est un polynôme de degré n
- $L_i(x_i) = 1$
- $L_i(x_j) = 0, \quad j \neq i$

Les polynômes de Lagrange forment une base (famille libre) de $\mathbb{R}_n[x]$ et par corollaire, on en déduit que tout polynôme de degré n s'écrit comme combinaison linéaire des L_i.
En particulier, interpoler la fonction f par un polynôme p de degré $\leq n$, aux $n+1$ points $x_0, x_1, \ldots, x_n$, revient à trouver p tel que:

$$p(x_i) = f(x_i) \quad \forall i \in \{0, \ldots, n\} \tag{3}$$

Si un tel polynôme existe, il s'écrit de manière unique:

$$p(x) = \sum_{i=0}^{n} \alpha_i L_i(x) \tag{4}$$

En prenant $x = x_j$, on a alors:

$$p(x_j) = \alpha_0 \underbrace{L_0(x_j)}_{0} + \cdots + \alpha_j \underbrace{L_j(x_j)}_{1} + \cdots + \alpha_n \underbrace{L_n(x_j)}_{0} = \alpha_j = f(x_j) \tag{5}$$

Par conséquent:

$$p(x) = \sum_{i=0}^{n} f(x_i) L_i(x) \tag{6}$$

L'exemple suivant (dont la solution $p(x) = x^2$ est connue) illustre l'interpolation polynomiale lagrangienne.

```
>>> from scipy import interpolate
... xi=np.array([0,1,2]); # n=2
... fi=xi**2;
```

$$\begin{array}{rcccl} L_0(x) & = & \dfrac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} & = & \dfrac{(x-1)(x-2)}{(0-1)(0-2)} & = & \dfrac{1}{2}x^2 - \dfrac{3}{2}x + 1 \\ L_1(x) & = & \dfrac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} & = & \dfrac{(x-0)(x-2)}{(1-0)(1-2)} & = & -x^2 + 2x \\ L_2(x) & = & \dfrac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} & = & \dfrac{(x-0)(x-1)}{(2-0)(2-1)} & = & \dfrac{1}{2}x^2 - \dfrac{1}{2}x \end{array} \tag{7}$$

Ce qui conduit bien entendu à:

$$p(x) = 0^2 L_0(x) + 1^2 L_1(x) + 2^2 L_2(x) = x^2 \tag{8}$$

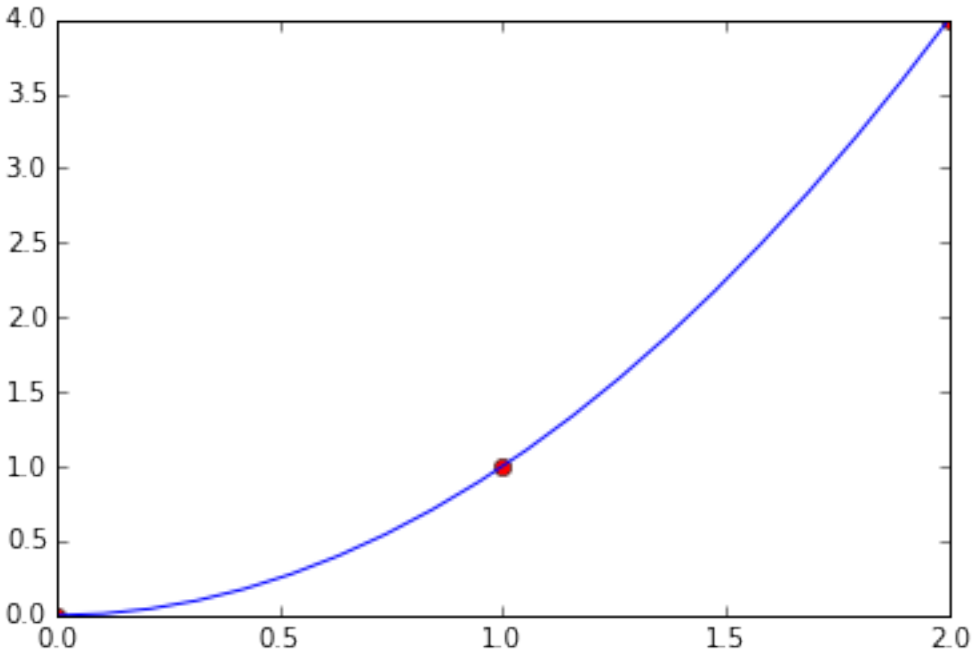
Ce résultat peut être trouvé en invoquant directement le module `interpolate`:

```
>>> p=sp.interpolate.lagrange(xi,fi)
... print p

2
1 x
```

La variable `p` est ici de type `poly1d` et peut donc être évaluée en invoquant `polyval`.

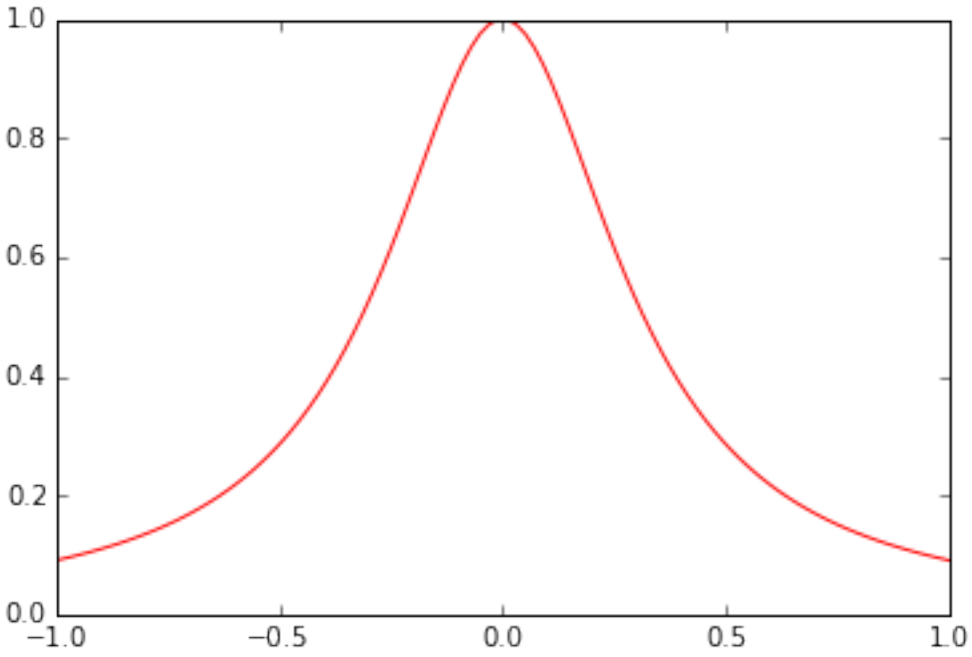
```
>>> fig, ax1= plt.subplots();
... ax1.plot(xi, fi,'ro');
... u=np.linspace(np.min(xi),np.max(xi),20);
... ax1.plot(u,np.polyval(p,u),'b-');
```



2.2 Phénomène de Runge

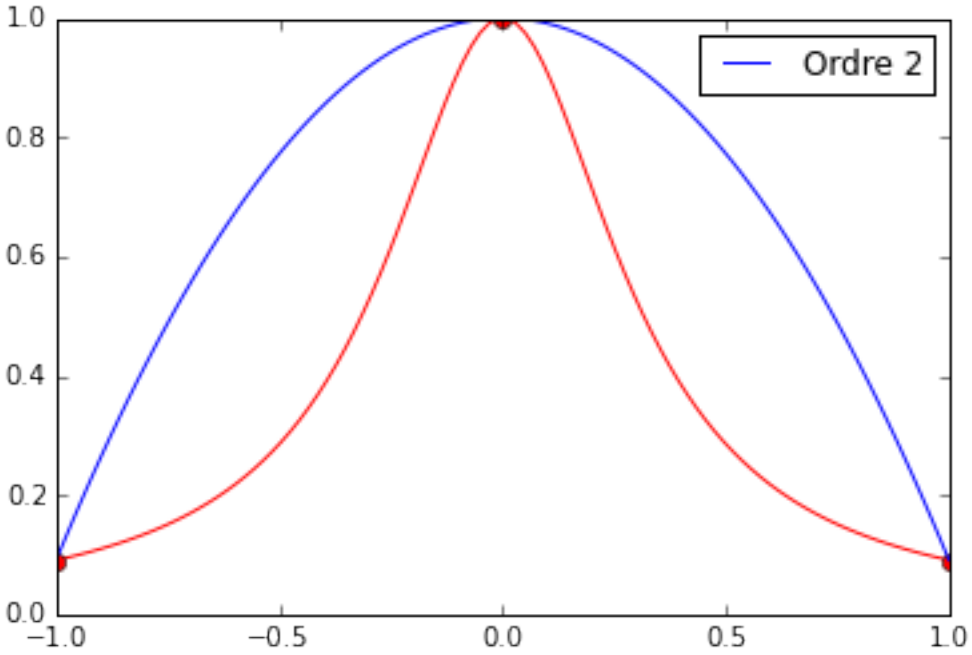
Prenons maintenant l'exemple de la function $x \mapsto 1/\left(1+10x^2\right)$ sur l'intervalle $[-1,1]$. Nous augmentons alors progressivement le degré d'interpolation.

```
>>> x=np.linspace(-1,1,100);
... f=1/(1+10*x**2);
... plt.plot(x, f,'r');
```



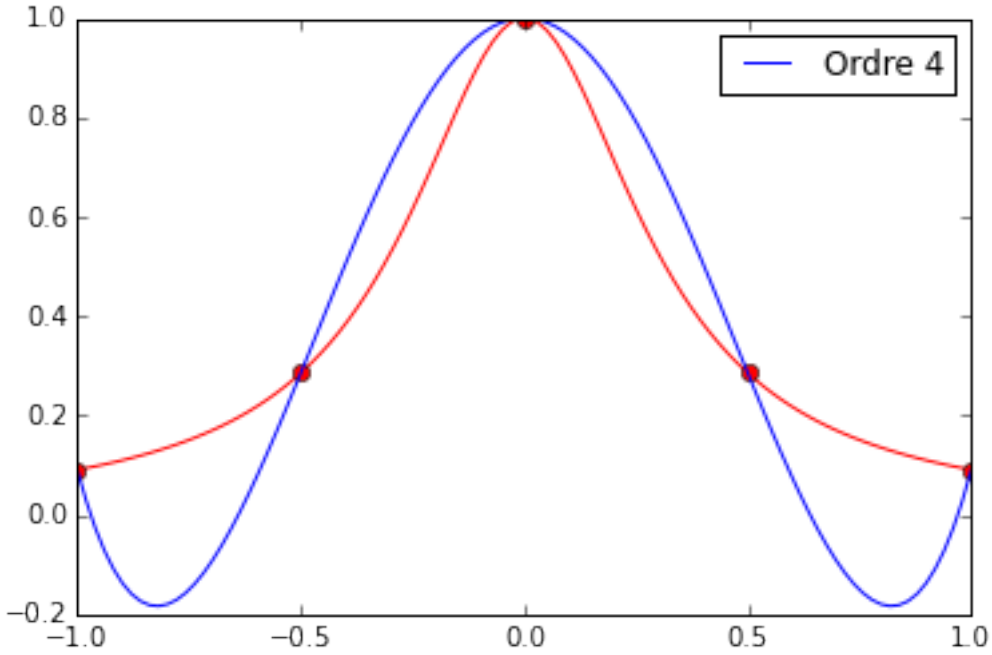
Interpolation d'ordre 2 à 3 points équidistants:

```
>>> xi=np.linspace(-1,1,3); # 3 points
... fi=1/(1+10*xi**2);
... p=sp.interpolate.lagrange(xi,fi);
... fig, ax1= plt.subplots();
... ax1.plot(x,f,'r');
... ax1.plot(xi,fi,'ro');
... ax1.plot(x,np.polyval(p,x),label='Ordre 2');
... ax1.legend();
```



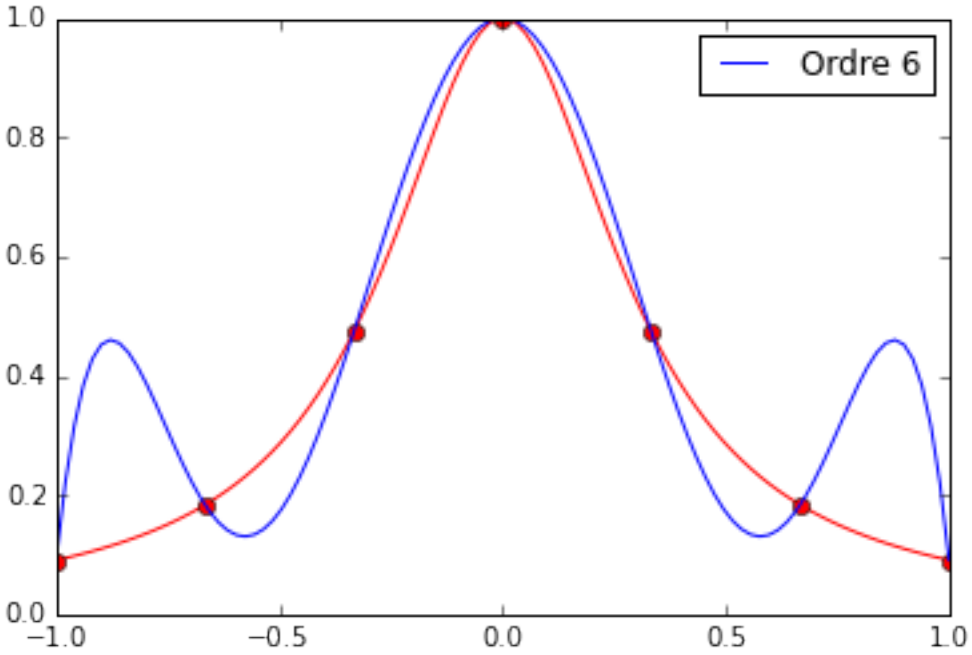
Interpolation d'ordre 4 à 5 points équidistants:

```
>>> xi=np.linspace(-1,1,5); # 5 points
... fi=1/(1+10*xi**2);
... p=sp.interpolate.lagrange(xi,fi);
... fig, ax1= plt.subplots();
... ax1.plot(x,f,'r');
... ax1.plot(xi,fi,'ro');
... ax1.plot(x,np.polyval(p,x),label='Ordre 4');
... ax1.legend();
```



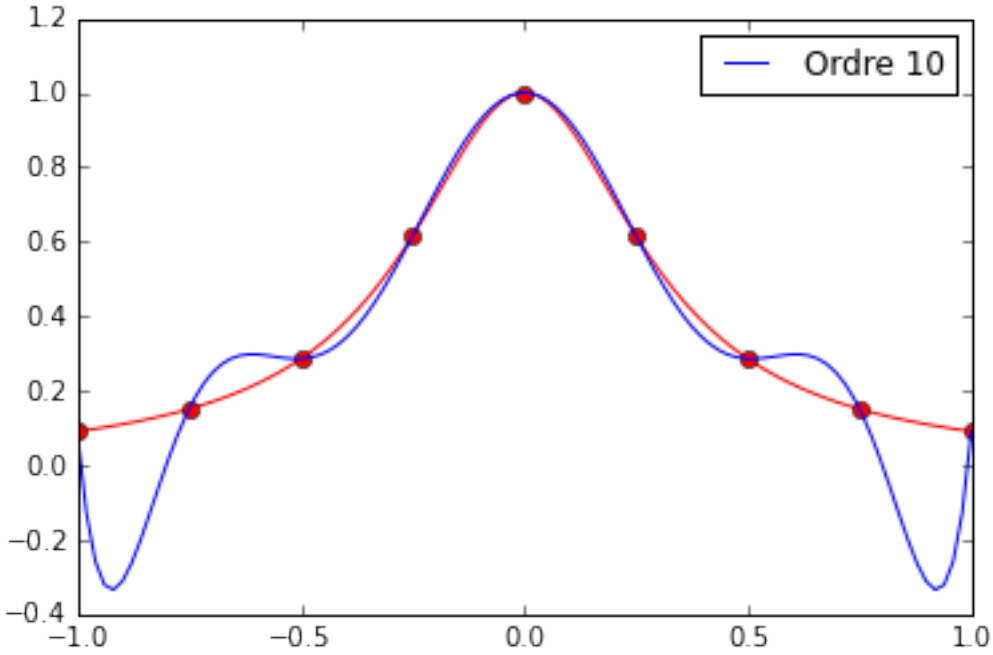
Interpolation d'ordre 6 à 7 points équidistants:

```
>>> xi=np.linspace(-1,1,7); # 7 points
... fi=1/(1+10*xi**2);
... p=sp.interpolate.lagrange(xi,fi);
... fig, ax1= plt.subplots();
... ax1.plot(x,f,'r');
... ax1.plot(xi,fi,'ro');
... ax1.plot(x,np.polyval(p,x),label='Ordre 6');
... ax1.legend();
```



Et enfin interpolation d'ordre 8 à 9 points équidistants:

```
>>> xi=np.linspace(-1,1,9); # 9 points
... fi=1/(1+10*xi**2);
... p=sp.interpolate.lagrange(xi,fi);
... fig, ax1= plt.subplots();
... ax1.plot(x,f,'r');
... ax1.plot(xi,fi,'ro');
... ax1.plot(x,np.polyval(p,x),label='Ordre 10');
... ax1.legend();
```



Un phénomène d'oscillation apparaît et s'amplifie lorsque n augmente. Ce phénomène a été clairement mis en évidence et expliqué par Carl Runge en 1901.

2.3 Polynômes de Chebyshev et atténuation du phénomène de Runge

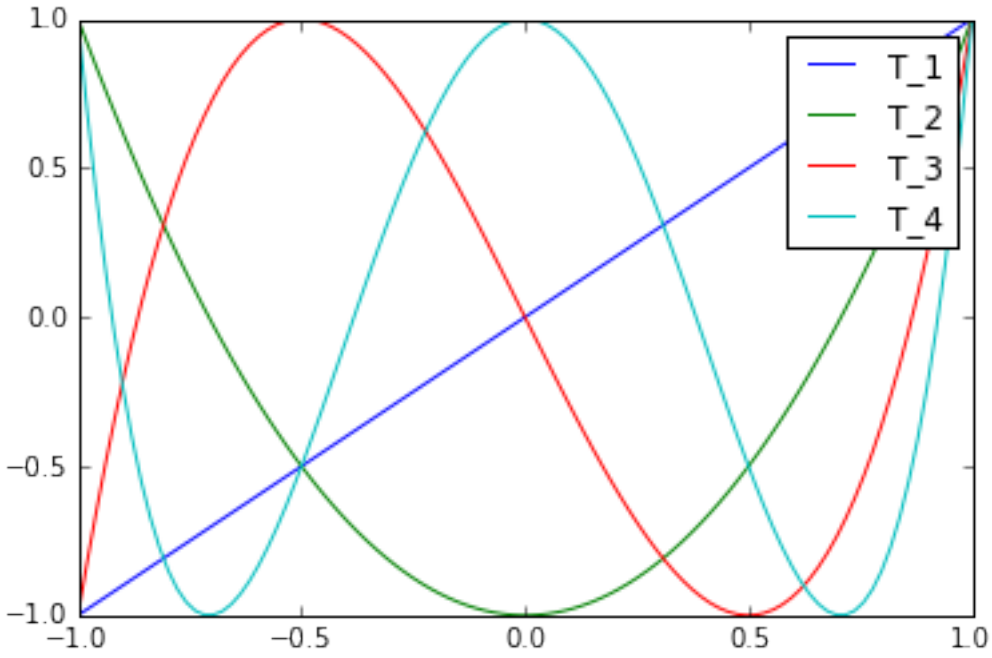
Le phénomène de Runge peut-être considérablement atténué en choisissant judicieusement les points d'évaluation. En particulier, on peut démontrer qu'en choisissant les racines des **polynômes de Chebyshev** (Tchebychev) comme points d'évaluation, on minimise les écarts entre la fonction interpolée et le polynôme d'interpolation.
On appelle polynôme de Chebyshev de degré n, le polynôme T_n défini sur $[-1, 1]$ par:

$$T_n(x) = \cos(n \arccos(x)) \tag{9}$$

Il est possible de tracer ces polynômes via le module `numpy.polynomial.chebyshev`.

```
>>> x=np.linspace(-1,1,100); fig, ax1=plt.subplots(); i=1; chebObj_=list(); chebPol_=list() # Init.
... while i<5: # Calcul des 4 premiers polynômes
...     chebOrd_ = np.array([0]*i+[1]);
...     chebObj_.append(np.polynomial.chebyshev.Chebyshev(chebOrd_));
...     chebPol_.append(np.polynomial.chebyshev.cheb2poly(chebObj_[-1].coef));
...     ax1.plot(x,np.polyval(chebPol_[-1][::-1],x),label='T_'+str(i));
```

```
...     i += 1;
... ax1.legend();
```



On obtient aisément les racines en utilisant la commande `roots`.

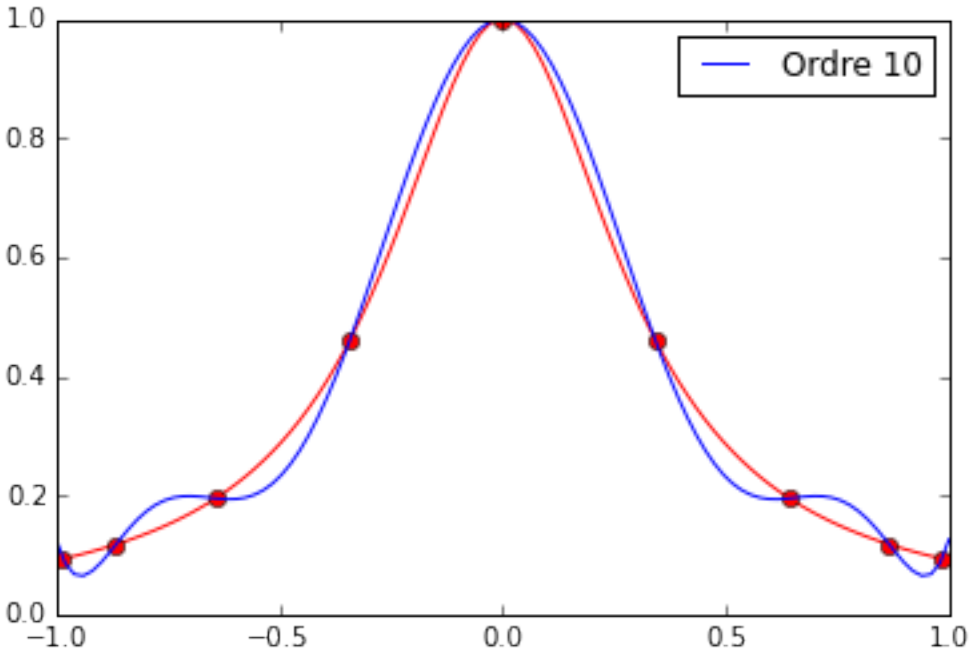
```
>>> for n, p in enumerate(chebObj_):
...     print n+1, p.roots()
```

```
1 [ 0.]
2 [-0.70710678  0.70710678]
3 [ -8.66025404e-01   2.81729083e-17   8.66025404e-01]
4 [-0.92387953 -0.38268343  0.38268343  0.92387953]
```

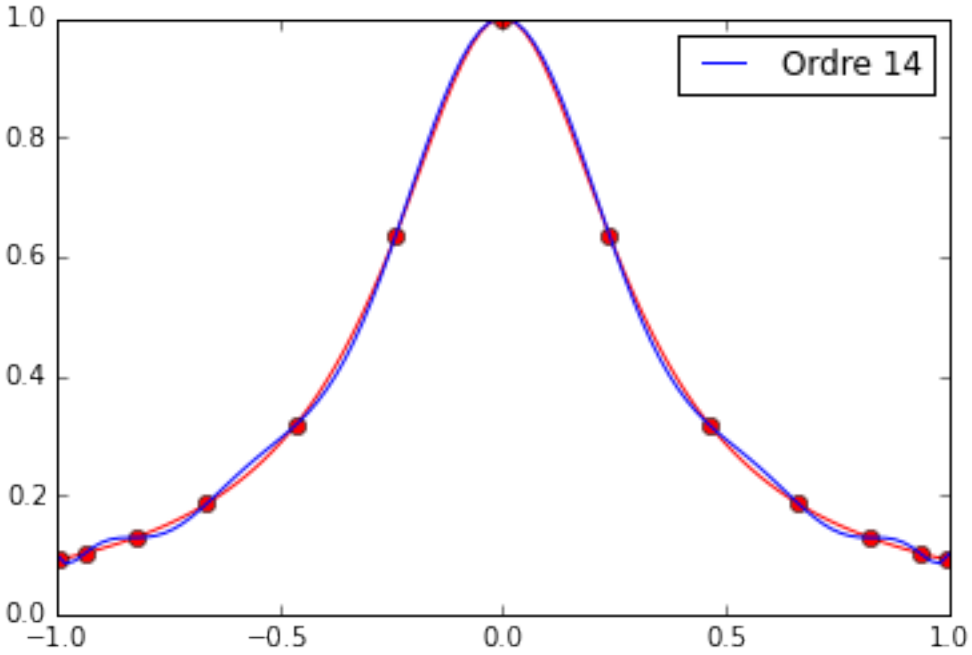
Ces valeurs peuvent alors être utilisées comme abscisses pour évaluer les polynômes de Lagrange et nous pouvons reprendre l'exemple de la fonction $x \mapsto 1/\left(1+10x^2\right)$ sur l'intervalle $[-1,1]$ pour illustrer l'atténuation du phénomène de Runge.

```
>>> x=np.linspace(-1,1,100);
... f=1/(1+10*x**2);
... xi=np.polynomial.chebyshev.Chebyshev(np.array([0]*9+[1])).roots(); # Ordre 10, 9 racines
... fi=1/(1+10*xi**2);
... p=sp.interpolate.lagrange(xi,fi);
```

```
>>> fig, ax1= plt.subplots();
... ax1.plot(x,f,'r');
... ax1.plot(xi,fi,'ro');
... ax1.plot(x,np.polyval(p,x),label='Ordre 10');
... ax1.legend();
```




```
>>> xi=np.polynomial.chebyshev.Chebyshev(np.array([0]*13+[1])).roots(); # Ordre 14, 13 racines
... fi=1/(1+10*xi**2);
... p=sp.interpolate.lagrange(xi,fi);
... fig, ax1= plt.subplots();
... ax1.plot(x,f,'r');
... ax1.plot(xi,fi,'ro');
... ax1.plot(x,np.polyval(p,x),label='Ordre '+str(len(xi)+1));
... ax1.legend();
```



2.4 Formulation newtonienne et différences divisées

Bien avant Lagrange, Newton fut un grand contributeur aux problèmes d'interpolation et écrivit dès 1676 (soit près de 120 ans avant Lagrange) une méthode "pour décrire une courbe géométrique qui doit passer par une série de points donnés". Il s'agit d'une interpolation polynomiale permettant d'obtenir uen interpolation de Lagrange (avant l'heure) comme une combinaison linéaire des polynômes de la **base newtonienne**.
Le probème de Newton est le suivant:
Soit $x_0, x_1, \ldots, x_n$, $n+1$ réels distincs et f une fonction continue sur une segment $[a, b]$. Newton cherche à trouver le polynôme p de degré $\leq n$ permettant d'interpoler f aux $n+1$ points x_i tel que:

$$\begin{cases} p(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_0 \\ p(x_i) = f(x_i) \quad \forall i \in \{0, \ldots, n\} \end{cases} \tag{10}$$

Prenons alors l'exemple d'un polynôme de degré 2 passant exactement par 3 points x_0, x_1, x_2.

$$\begin{cases} p(x) = \alpha x^2 + \beta x + \gamma \\ p(x_0) = \alpha x_0^2 + \beta x_0 + \gamma = f(x_0) = y_0 \\ p(x_1) = \alpha x_1^2 + \beta x_1 + \gamma = f(x_1) = y_1 \\ p(x_2) = \alpha x_2^2 + \beta x_2 + \gamma = f(x_2) = y_2 \end{cases} \tag{11}$$

L'élimination de γ s'obtient par les différences suivantes:

$$\begin{cases} y_1 - y_0 = (x_1 - x_0)\left[\alpha\left(x_1 + x_0\right) + \beta\right] \Rightarrow \dfrac{y_1 - y_0}{x_1 - x_0} = \alpha\left(x_1 + x_0\right) + \beta \\ y_2 - y_1 = (x_2 - x_1)\left[\alpha\left(x_2 + x_1\right) + \beta\right] \Rightarrow \dfrac{y_2 - y_1}{x_2 - x_1} = \alpha\left(x_2 + x_1\right) + \beta \end{cases} \tag{12}$$

Nous procédons de même pour l'élimination de β:

$$\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0} = \alpha\left(x_2 - x_0\right) \tag{13}$$

A l'ordre 1, la fonction f peut alors être approchée par:

$$p\left(x\right) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}\left(x - x_0\right) \tag{14}$$

Puis à l'ordre 2 par:

$$p\left(x\right) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}\left(x - x_0\right) + \frac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}\left(x - x_0\right)\left(x - x_1\right) \tag{15}$$

Et utilisant les notations des **différences divisées** ...

$$\begin{array}{lll} \bullet & y_0 & = \quad y\left[x_0\right] \\ \bullet & \dfrac{y_1 - y_0}{x_1 - x_0} & = \quad y\left[x_0, x_1\right] \\ \bullet & \dfrac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} = \dfrac{y\left[x_1, x_2\right] - y\left[x_0, x_1\right]}{x_2 - x_0} & = \quad y\left[x_0, x_1, x_2\right] \end{array} \tag{16}$$

... nous obtenons:

$$p\left(x\right) = y\left[x_0\right] + y\left[x_0, x_1\right]\left(x - x_0\right) + y\left[x_0, x_1, x_2\right]\left(x - x_0\right)\left(x - x_1\right) \tag{17}$$

Les points $\left(0, 0^2\right)$, $\left(1, 1^2\right)$ et $\left(2, 2^2\right)$ de l'exemple sur la formulation lagrangienne peuvent être repris pour construire le tableau suivant:

$$\begin{array}{cccc} x_i & y_i & y\left[x_i, x_j\right] & y\left[x_i, x_j, x_k\right] \\ 0 & 0 & & \\ & & 1 & \\ 1 & 1 & & 1 \\ & & 3 & \\ 2 & 4 & & \end{array} \tag{18}$$

Ce qui conduit bien à:

$$p\left(x\right) = 0 + \left(x - 0\right) + 1\left(x - 0\right)\left(x - 1\right) = x^2 \tag{19}$$

Les polynômes de Newton forment une base de $\mathbb{R}_n\left[x\right]$ et s'écrivent:

$$N_i(x) = \prod_{j=0}^{i-1}(x - x_j) = (x - x_0)(x - x_1)\dots(x - x_{i-1}) \tag{20}$$

La formulation newtonnienne revient alors à:

$$p(x) = \sum_{i=0}^{n} y[x_0, \dots, x_i] N_i(x) \tag{21}$$

L'avantage d'une telle formulation sur une formulation lagrangienne réside dans le fait que la base des polynômes n'a pas besoin d'être r éécrite lorsque l'on souhaite ajouter un point.
Le nombre d'opérations nécessaires dans une formulation newtonienne est par ailleurs très inférieure $O(n)$ à celles requises par la formulation lagrangienne $O\left(n^2\right)$. La notation O est ici utilisé pour mesurer la **complexité algorithmique**.

2.5 Formulation barycentrique

La formulation barycentrique provient d'une réécriture de la formulation lagrangienne afin de diminuer le nombre d'opérations à réaliser.
C'est à dire passer de $O\left(n^2\right)$ opérations à $O(n)$.
Les polynômes de Lagrange peuvent s'écrire sous la forme suivante:

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{1}{x - x_i} \prod_{j=0}^{n}(x - x_j) \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{1}{x_i - x_j} \tag{22}$$

Ce qui permet de réécrire le polynôme d'interpolation:

$$p(x) = \sum_{i=0}^{n} f(x_i) L_i(x) = \prod_{j=0}^{n}(x - x_j) \sum_{i=0}^{n} \left[\frac{f(x_i)}{x - x_i} \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{1}{x_i - x_j} \right] \tag{23}$$

Cette première forme (Rutishauser, 1990) peut encore être améliorée en posant:

$$\lambda_i = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{1}{x_i - x_j} \tag{24}$$

Par ailleurs, si f est constante et égale à 1 alors la seule solution possible est $p(x) = 1$, ce qui conduit à la deuxième forme véritablement appelée **formulation barycentrique** (Rutishauser, 1990):

$$1 = \prod_{j=0}^{n}(x - x_j) \sum_{i=0}^{n} \left[\frac{\lambda_i}{x - x_i} \right] \Rightarrow p(x) = \sum_{i=0}^{n} \left[\frac{\frac{\lambda_i}{x - x_i} f(x_i)}{\frac{\lambda_i}{x - x_i}} \right] \tag{25}$$

Au moins deux avantages de la formulation barycentrique sur la formulation newtonienne peuvent êre évoqués: Grâce à la formulation barycentrique,

- les quantités qui doivent être évaluées de manière récursive ne dépendent pas de f (contrairement aux différences finies de la formulation newtonienne. Ue fois les poids λ_i connus, n'importe quelle fonction peut-être interpolée.
- les caluls sont indépendants de l'ordre des points.

L'un des avantages de la formulation newtonienne réside cependant dans la facilité d'incorporer des contraintes supplémentaires sur les dérivées comme pour l'interpolation d'Hermite. Il existe également quelques domaines où l'utilisation de la formulation newtonienne est préférable.

2.6 Formulation d'Hermite, prise en compte des dérivées

Une extension possible des interpolations précédentes repose sur la prise en considération de conraintes supplémentaires relatives aux dérivées de la fonction. En particulier, il peut être recherché que le polynôme p soit à la fois interpolateur et osculateur de la fonction f. C'est à dire que la courbe représentative du polynôme passe par les points choisis (**polynôme interpolateur**) avec une courbure pilotée par la valeur de la dérivée de la fonction en ces points (**polynôme osculateur**).