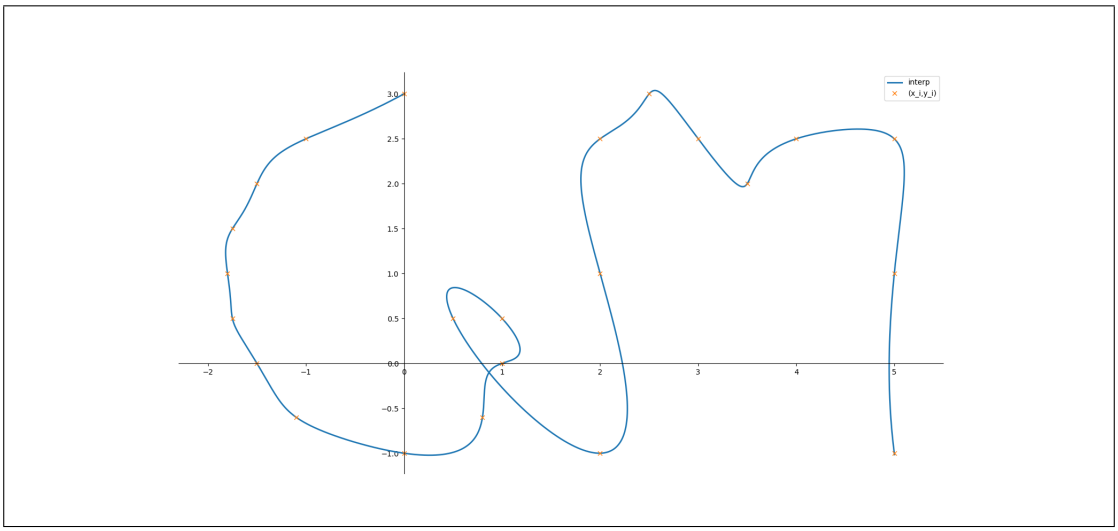
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

GM3
2022/2023

TP1 : Interpolation polynomiale et dessin

Analyse Numérique 2 - MMSN - GM3



Etudiants : Guines Antoine, Langolff Clément

Contents

I. Interpolation d'une fonction $f(x)$ **3**

1. Théorie **4**
1.1. Évaluation du polynôme d'interpolation en x 4
1.2. Interpolation d'Hermite 5
1.2.1. Degré du polynôme d'interpolation 5
1.2.2. Base de Newton 6
1.2.3. Algorithme des différences divisées pour l'interpolation d'Hermite . 7

2. Code **10**
2.1. Doit-on adapter Hörner ? 10
2.2. Comparaison des méthodes implémentées 10
2.2.1. Étude de l'influence de la répartition des points d'interpolation . . 10
2.2.2. Étude de l'influence du nombre de points d'interpolation 13
2.2.3. Étude de l'influence de la méthode d'évaluation utilisée 19
2.2.4. Etude du phénomène Runge 20
2.2.5. Bonus 1 21
2.2.6. Bonus 2 21

II. Interpolation de la trajectoire de Moustik **22**

3. Aspect théorique **23**

4. Code **25**
4.1. Test des différentes méthodes sur les jeux de données 25
4.2. Nouveau jeu de données (Bonus) 28

Conclusion **29**

Part I.

Interpolation d'une fonction $f(x)$

1. Théorie

1.1. Évaluation du polynôme d'interpolation en x

L'évaluation du polynôme d'interpolation en un point $x \neq x_i$ s'effectue en 2 étapes :

- Le calcul des coefficients de la décomposition du polynôme d'interpolation dans la base de Newton grâce à l'algorithme des différences divisées.
- L'application de l'algorithme d'Hörner au polynôme d'interpolation, nous permettant d'obtenir sa valeur en x.

Plus précisément on commence tout d'abord par définir la base de Newton, qui est la famille de fonction $(N_i)_{i=\{0,...,n\}}$ telle que :

$$\begin{cases} N_0(x) = 1 \\ N_i(x) = \prod_{j=0}^{i-1}(x - x_j) \end{cases}$$

Cette famille de fonctions forme une base de $\mathbb{P}_n$ et vérifie de plus $N_i(x_j) = 0 \ \forall j < i$. Ainsi le polynôme d'interpolation $P(x)$ passant par les $n+1$ points peut s'écrire sous la forme :

$$P(x) = \sum_{j=0}^{n} \beta_j N_j(x)$$

Pour pouvoir évaluer ce polynôme en x, il nous faut naturellement trouver l'expression des β_j. C'est dans ce but qu'intervient la formule des différences divisées. On pose alors :

$$f[x_i, ..., x_{i+j+1}] = \frac{f[x_{i+1}, ..., x_{i+j+1}] - f[x_i, ..., x_{i+j}]}{x_{i+j+1} - x_i} \tag{1.1}$$

et

$$f[x_i] = f(x_i) = y_i$$

On démontre *(cf proposition 2.2)* alors que le polynôme d'interpolation de degré n passant par les $n+1$ points $(x_i, y_i)_{i=\{0,...,n\}}$, est donné par :

$$P(x) = f[x_0]N_0(x) + f[x_0, x_1]N_1(x) + ... + f[x_0, x_1, ..., x_n]N_n(x)$$

On remarque donc que l'on a :

$$\beta_j = f[x_0, x_1, ..., x_j] \ \ \forall j \in \{0, ..., n\}$$

Ainsi on calcul par l'algorithme des différences divisées les coefficients du polynôme d'interpolation en utilisant la formule 1.1. Cette étape n'est à faire qu'une fois car les β_j

ne dépendent pas du point d'évaluation de P. Donc peu importe le point d'évaluation choisi, la décomposition restera identique. Ce qui va changer en revanche, c'est l'algorithme d'Hörner que l'on va maintenant expliquer.
Évaluer P en α revient à écrire :

$$P(\alpha) = \sum_{i=0}^{n} \beta_i N_i(\alpha)$$

De plus on a par définition :

$$N_{i+1}(\alpha) = \prod_{j=0}^{i} (\alpha - x_j) = (\alpha - x_i) N_i(\alpha)$$

Ainsi cela nous permet d'écrire $P(\alpha)$ sous la forme :

$$P(\alpha) = \underbrace{\beta_0 + (\alpha - x_0)(\beta_1 + (\alpha - x_1)(... \underbrace{(\beta_{n-1} + (\alpha - x_{n-1}) \underbrace{\beta_n}_{b_n})}_{b_{n-1}}))}_{b_0}$$

On va donc commencer par évaluer le terme le plus à l'intérieur du produit puis on élargit progressivement, on pose donc :

$$\begin{cases} b_n = \beta_n \\ b_k = \beta_k + (\alpha - x_k) b_{k+1} \ \ \forall k \in \{0, ..., n-1\} \end{cases}$$

Ainsi, il vient que $P(\alpha) = b_0$ obtenu en partant de b_n et en appliquant successivement la relation de récurrence donnée ci-dessus. C'est exactement ce que fait l'*Algorithme 1* donné en fin de TP. Celui-ci initialise n avec le nombre de coefficient dans le tableau *coeff* qui correspond au nombre de β_j, puis applique la relation de récurrence ci-dessus pour trouver b_0.

Algorithme 1.1 Horner

$b \leftarrow \beta[n]$
Pour k de $n-1$ à 0
  $b \leftarrow \beta[k] + (A - x_k)b$
Retourner b

Remarque : Dans l'algorithme, A désigne un point d'évaluation différent des x_k.

1.2. Interpolation d'Hermite

1.2.1. Degré du polynôme d'interpolation

On cherche ici un polynôme d'interpolation pour approcher une fonction f. On utilise l'interpolation d'Hermite et dans note cas on demande seulement que :

$$\forall i \in \{0, ..., n\}, \ P^{(k)}(x_i) = f^{(k)}(x_i) \ \textit{pour} \ k = 0, 1$$

P se voit donc imposer $(n+1)$ conditions pour $k=0$ et pareil pour $k=1$, soit $2\times(n+1)$ conditions. On en déduit que :

$$deg(P) = 2n+2-1$$
$$deg(P) = 2n+1$$

Plus précisément on peut montrer que qu'il existe un unique polynôme $P \in P[X^N]$, avec dans notre cas $N=(n+1)\times 2-1$, qui soit solution du problème d'interpolation d'Hermite.
Preuve :
Montrons que si le polynôme P de degrès $N=(n+1)\times 2-1$ existe et est solution du problème d'interpolation d'Hermite, alors cette solution est unique.

Soit P et $Q \in P[X^N]$, deux solutions au problème d'interpolation d'Hermite. Posons $R=P-Q$, alors $R \in P[X^N]$ donc le degré de R est au plus N, et tous les x_i sont racines doubles de R. On a donc :

$$\underbrace{R(x)}_{deg=N=2\times(n+1)-1} = C(x) \times \underbrace{\prod_{i=0}^{n}(x-x_i)^2}_{deg=2\times(n+1)}$$

R possède alors plus de racines que son degré, donc $R(x)=0\ \forall x \in \mathbb{R}$ et ainsi $P=Q$. S'il existe une solution au problème alors celle-ci est unique.
Montrons maintenant que cette solution existe. On cherche donc $P \in P[X^N]$ vérifiant les $N+1$ équations, c'est à dire :

$$(\sum_{j=0}^{N}\alpha_j x_i^j)^{(k)}(x_i) = f^{(k)}(x_i)\ \ \forall i \in \{0,...,n\},\ k=0,1$$

$$\underbrace{\sum_{j=0}^{N}(x^j)^{(k)}(x_i)}_{M_{lj}} \times \alpha_j = f^{(k)}(x_i)\ \ \forall i \in \{0,...,n\},\ k=0,1$$

avec $l=i+(n+1)\times k$. Donc ce problème revient à résoudre un système linéaire de la forme :

$$M \times \underline{\alpha} = F$$

Avec M une matrice carré. Par unicité de la solution, M est injective, donc par le théorème du rang on a bien existence de la solution.

1.2.2. Base de Newton

Pour adapter le problème d'interpolation à des degrès supérieur, on procède à des "copie" de points. Pour le problème d'interpolation d'Hermite de degré 1, on dédouble les points. Cela signifie que l'on a maintenant :

$$x_0 = y_0 = y_1 < x_1 = y_2 = y_3 < ... < x_n = y_{2n} = y_{2n+1}$$

On a donc bien $2n+2$ points y_j ce qui est cohérent avec le degré du polynôme d'interpolation qui vaut $(2n+2)-1$. Ainsi la base de Newton obtenue à partir de ces points est définie telle que :

$$\begin{cases} N_0(x) = 1 \\ N_i(x) = \prod_{j=0}^{i-1}(x - y_j) \quad \forall i \in \{0, ..., 2n+1\} \end{cases}$$

Cette famille forme une base de $P[X^{2n+1}]$. Remarquons d'abord que $N_i(x) = 0$ $j > i$.

En effet soit $j > i$, on a :

$$N_i(x_j) = \prod_{k=0}^{i-1} \underbrace{(x_j - y_k)}_{0 \; quand \; k=j} = 0$$

Montrons que cette famille est libre. On pose :

$$Q(x) = \sum_{i=0}^{n} N_i(x)\alpha_i$$

Montrons que $Q(x) = 0 \; \forall x \iff \alpha_i = 0 \; \forall i \in \{0, ..., 2n+1\}$

Si on a $\sum_{i=0}^{n} N_i(x)\alpha_i = 0 \; \forall x$ alors en $x = y_0$:

$$\sum_{i=0}^{n} N_i(y_0)\alpha_i = 0 \iff \alpha_0 = 0$$

De même en $x = y_1$:

$$\sum_{i=0}^{n} N_i(y_1)\alpha_i = \alpha_1 \underbrace{N_1(y_1)}_{\neq 0} = 0 \iff \alpha_1 = 0$$

Et on continue ainsi pour déduire que $\alpha_i = 0 \; \forall i$.

Enfin, la famille des $(N_i(x))_{i \in 0,..,2n+1}$ possède $2n+2$ élèments et $dim(P[X^{2n+1}]) = 2n+2$. On déduit que la famille des $(N_i(x))_{i \in 0,..,2n+1}$ forme bien une base de $P[X^{2n+1}]$.

1.2.3. Algorithme des différences divisées pour l'interpolation d'Hermite

Dans le cours nous avons pu généraliser l'algorithme de Neville-Aitkens pour l'interpolation d'Hermite grâce au *Théorème 4.4* qui nous donne la relation suivante entre les T_k^i :

$$T_{k+1}^i(x) = \begin{cases} \frac{(y_{i+k+1}-x)T_k^i(x) - (y_i - x)T_k^{i+1}(x)}{(y_{i+k+1} - y_i)} & si \; y_{i+k+1} \neq y_i \\ T_k^i(x) + f^{'}(y_i)(x - y_i) & sinon \end{cases} \tag{1.2}$$

Les T_k^i sont les polynômes d'interpolation d'Hermite de degré k vérifiant la relation suivante :

$$(T_k^i(y_j))^{(l)} = f^{(l)}(y_j) \quad \forall j \in \{i, ..., i+k\}, \quad \forall l \in \{0, 1\}$$

Nous allons maintenant utiliser ces relations pour généraliser l'algorithme des différences divisées à un polynôme d'interpolation d'Hermite.
On va procéder de la même manière que pour démontrer l'algorithme classique des différences divisées.
On pose cette fois :

$$f[y_i, ..., y_{i+j}] = \begin{cases} \frac{f[y_{i+1},...,y_{i+j}]-f[y_i,...,y_{i+j-1}]}{y_{i+j}-y_i} & si\ y_{i+j} \neq y_i \\ \frac{f^{(j)}(y_i)}{j!} & si\ y_{i+j} = y_i \end{cases}$$

Ceci est la formule générale pour des ordres de dérivations supérieurs à 1. Mais dans notre cas comme nous n'avons fait que dédoubler les points, on a $y_{i+j} = y_i$ seulement si $j = 1$. On peut donc réécrire que dans le cas $y_{i+j} = y_i$ (i.e. $j = 1$), on a $f[y_i, ..., y_{i+j}] = f'(y_i)$. De plus, on sait d'après la question précédente que notre polynôme d'interpolation admet une décomposition dans la base de Newton telle que :

$$P(x) = \sum_{j=0}^{2n+1} \beta_j N_j(x) = T_{2n+1}^0(x)$$

On veut montrer que cette décomposition s'écrit en fait comme suit :

$$P(x) = f[y_0]N_0(x) + f[y_0, y_1]N_1(x) + ... + f[y_0, y_1, ..., y_{2n+1}]N_{2n+1}(x)$$

Preuve :

Soit T_k^i le polynôme d'interpolation d'Hermite définit précédemment (1.2). Prouvons par récurrence sur k que $\forall i \in \{0, ..., 2n+1\}$ on ait :

$$T_k^i(x) = f[y_i] + f[y_i, y_{i+1}](x - y_i) + ... + f[y_i, y_{i+1}, ..., y_{i+k}] \prod_{j=i}^{i+k-1} (x - y_j) \qquad (1.3)$$

Initialisation : Pour $k = 0$, la proposition est vraie car $f[y_i] = f(y_i) = T_0^i(x)$.
Hérédité : Supposons la propriété vraie au rang k, et montrons son hérédité. On peut donc écrire T_{k+1}^i dans la base de Newton comme suit :

$$T_{k+1}^i(x) = \beta_0 + \beta_1(x - y_i) + ... + \beta_{k+1} \prod_{j=i}^{i+k} (x - y_j)$$

On évalue alors $T_k^i(x)$ et $T_{k+1}^i(x)$ en $x = y_i$, ce qui nous donne : $\begin{cases} T_k^i(y_i) = f[y_i] \\ T_{k+1}^i(y_i) = \beta_0 \end{cases}$. Or $\forall j \in \{i, ..., i+k\}$, on a $T_k^i(y_j) = T_{k+1}^i(y_j)$. On déduit alors que $\beta_0 = f[y_i]$. De la même façon on obtient $T_k^i(y_{i+1}) = f[y_i] + f[y_i, y_{i+1}](y_{i+1} - y_i)$ et $T_{k+1}^i(y_{i+1}) = f[y_i] + \beta_1(y_{i+1} - y_i)$, on déduit donc que $\beta_1 = f[y_i, y_{i+1}]$. On répète le même processus $\forall y_j, j \in \{i, ..., i+k\}$.

Il nous reste maintenant à montrer que : $\beta_{k+1} = f[y_i, y_{i+1}, ..., y_{i+k+1}]$. Pour cela on utilise la relation (1.2). De plus on peut écrire que

$$T^i_{k+1}(x) = T^i_k(x) + \beta_{k+1} \prod_{j=i}^{i+k} (x - y_j)$$

Donc β_{k+1} est facteur du terme de plus haut degré. On distingue alors 2 cas suivant si $y_i \neq y_{i+k+1}$ ou si $y_i = y_{i+k+1}$.

- Si $y_i \neq y_{i+k+1}$ alors on utilise la formule associée de la relation (1.2), on remplace T^i_k et T^{i+1}_k par l'hypothèse de récurrence (1.3), et on regarde uniquement le coefficient de plus haut degré, on a alors :

$$\beta_{k+1} = \frac{f[y_{i+1}, ..., y_{i+k+1}] - f[y_i, y_{i+1}, ..., y_{i+k}]}{y_{i+k+1} - y_i} = f[y_i, y_{i+1}, ..., y_{i+k+1}]$$

- Si $y_i = y_{i+k+1}$, alors comme on a juste dédoublé les points x_i cela n'arrive que si $k = 0$ (i.e pour calculer la deuxième colonne du tableau). Si on imposait une égalité de P avec f à des ordres de dérivations supérieures, cela vaudrait également pour d'autres k (i.e pour d'autre colonnes). Dans notre cas on obtient directement que si $y_i = y_{i+1}$ on a par la relation (1.2) :

$$\beta_{k+1} = \beta_1 = f'(y_i)$$

Ce qui prouve que l'on peut bien écrire :

$$T^i_k(x) = f[y_i] + f[y_i, y_{i+1}](x - y_i) + ... + f[y_i, y_{i+1}, ..., y_{i+k}] \prod_{j=i}^{i+k-1} (x - y_j) \;\; \forall k \in \{0, ..., i\}, \;\; \forall i \in \{0, ..., 2n+1\}$$

Par unicité du polynôme d'interpolation, $T^0_{2n+1}(x) = P(x)$, on déduit que l'on peut écrire P tel que :

$$P(x) = f[y_0]N_0(x) + f[y_0, y_1]N_1(x) + ... + f[y_0, y_1, ..., y_{2n+1}]N_{2n+1}(x)$$

Et cela justifie alors la formule posée pour déterminer les $f[y_i, ..., y_{i+j}]$.

2. Code

2.1. Doit-on adapter Hörner ?

Non, il n'est pas nécessaire de changer l'algorithme de Hörner pour évaluer le polynôme d'interpolation en un point x. En effet, comme expliquer dans la section 1.1, l'algorithme de Hörner se construit à partir de la décomposition du polynôme d'interpolation dans la base de Newton. Or pour l'interpolation d'Hermite on dédouble les x_i en y_j de la sorte : $x_0 = y_0 = y_1 < x_1 = y_2 = y_3 < ... < x_n = y_{2n} = y_{2n+1}$. Les points que l'on va maintenant considérer sont donc les $(y_i)_{i=0,..,2n+1}$. Et le fait qu'il soit 2 à 2 égaux ne changent en rien Hörner. En effet le polynôme d'interpolation d'Hermite associé à ces y_i admet lui aussi une décomposition dans la base de Newton tel que :

$$\begin{aligned} P(x) &= f[y_0]N_0(x) + f[y_0, y_1]N_1(x) + ... + f[y_0, y_1, ..., y_{2n+1}]N_{2n+1}(x) \\ &= \sum_{j=0}^{2n+1} f[y_0, ..., y_j]N_j(x) \end{aligned}$$

avec cette fois : $N_i(x) = \prod_{j=0}^{i-1}(x - y_j)$. Donc on peut refaire exactement comme dans la partie théorique, simplement dans notre cas les β_j sont les $f[y_0, ..., y_j]$, il y a donc 2 fois plus de coefficients que dans le cas classique mais la manière de procéder est strictement la même. C'est à dire qu'on aura évalué en α :

$$\begin{cases} b_{2n+1} = \beta_{2n+1} = f[y_0, y_1, ..., y_{2n+1}] \\ b_k = \beta_k + (\alpha - x_k)b_{k+1} \quad \forall k \in \{0, ..., 2n\} \end{cases}$$

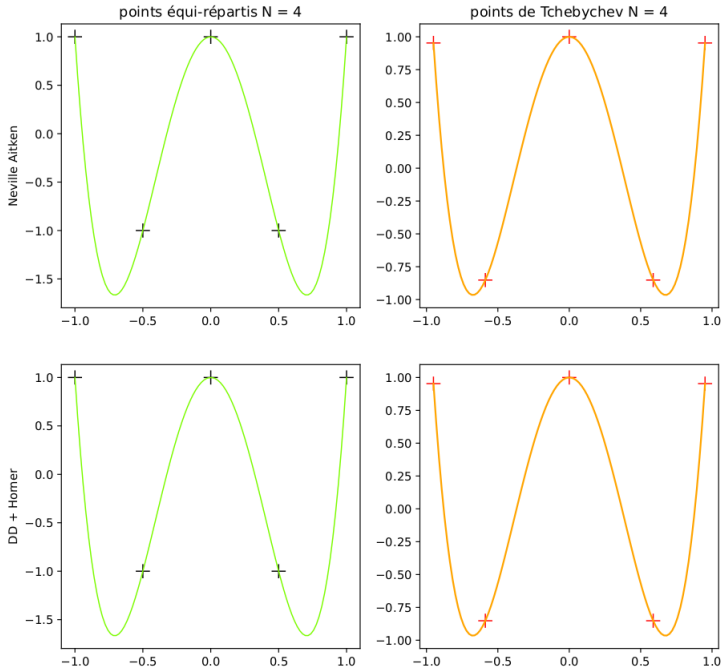
Et on descendra jusqu'à obtenir $P(\alpha) = b_0$.

2.2. Comparaison des méthodes implémentées

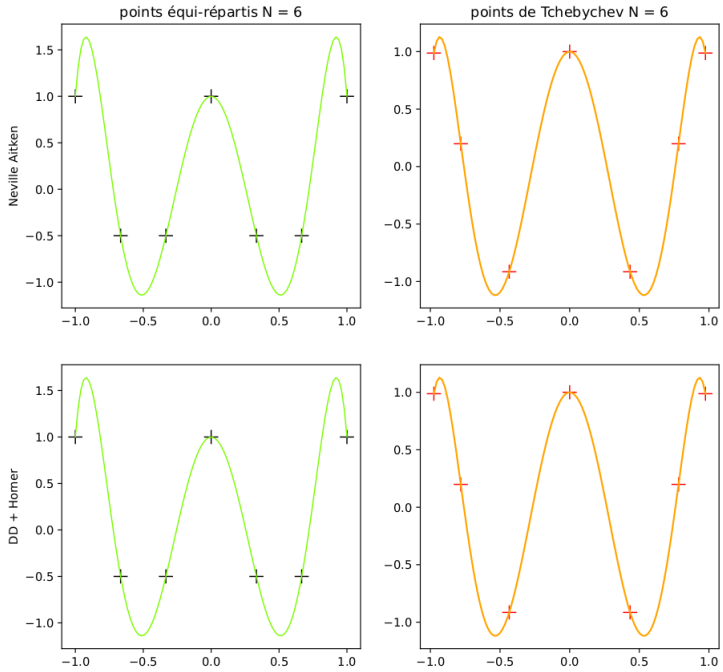
2.2.1. Étude de l'influence de la répartition des points d'interpolation

Dans cette partie, nous allons procéder à la comparaison de 2 méthodes différentes pour le choix du positionnement de nos points d'interpolation. La première méthode consiste à choisir des points équi-répartis, la seconde à prendre les points de Tchebychev, qui sont plus exactement les racines du polynôme de Tchebychev $T_{n+1}(x)$, polynôme de degré $n + 1$. Observons l'allure des polynômes d'interpolation suivant le choix des points. Prenons dans un premier temps $N = 4$, c'est à dire 5 points d'interpolation. On a les

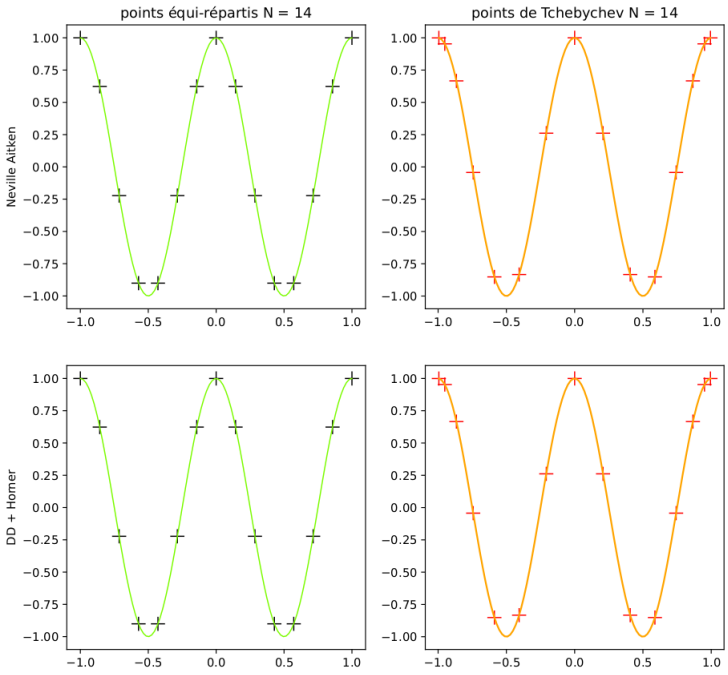
graphes suivants :



On remarque immédiatement que pour les 2 algorithmes utilisés les points de Tchebychev donnent une bien meilleure approximation de la fonction interpolée, qui est ici $f(x) = cos(2\pi x)$. En effet le polynôme d'interpolation obtenu avec les points équi-répartis est relativement éloigné de $f(x)$ puisqu'il possède un minimum inférieur à -1.5, alors que $f(x) \in [-1; 1]$. De plus ces minima devraient être situé en $-\frac{1}{2}$ et $\frac{1}{2}$, et sont ici plutôt aux alentours de $0,75$. Le polynôme obtenu grâce aux points d'interpolation de Tchebychev, lui est bien plus proche de $f(x)$, bien que ses minima soient également atteint en $\pm 0,7$ au lieu de $\pm 0,5$.
De même pour $N = 6$ on obtient les graphes suivant :



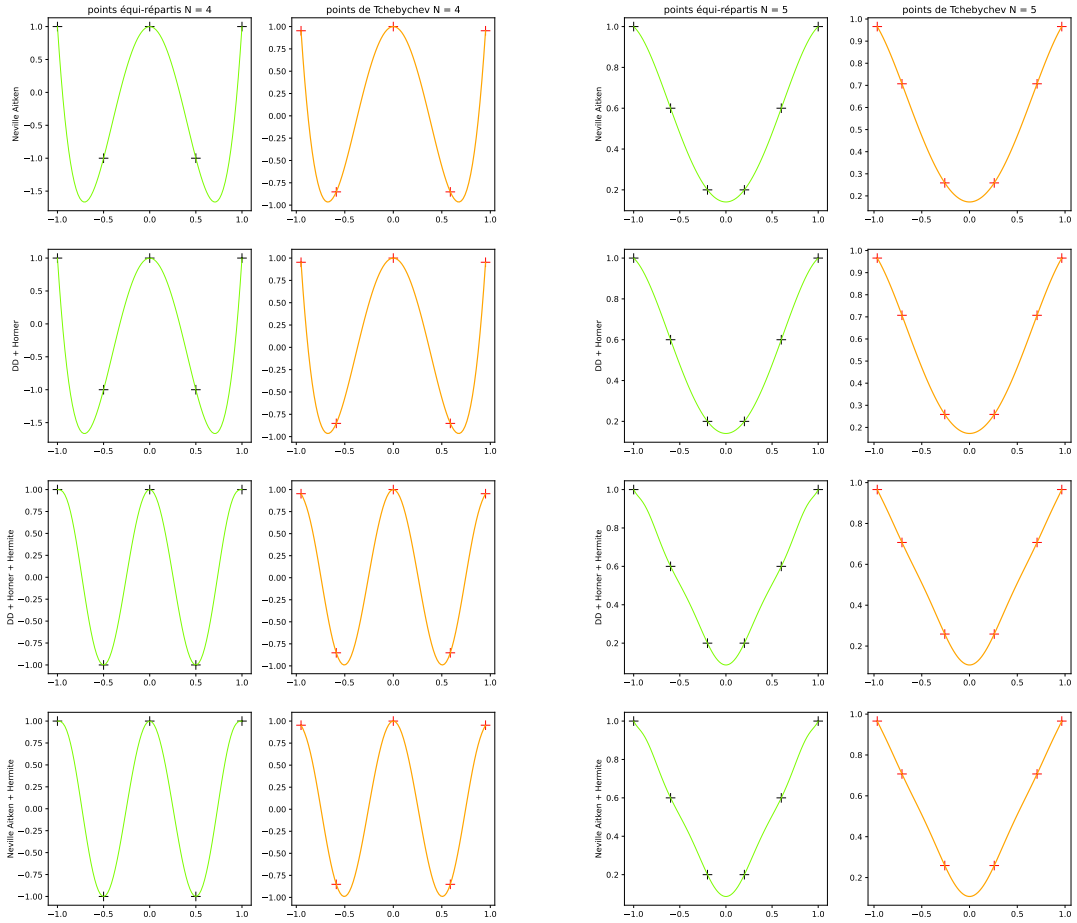
Comme précédemment le polynôme d'interpolation obtenu avec les points équi-répartis est moins bon que celui obtenu à partir des points de Tchebychev. On peut conclure que lorsque le nombre de points est relativement faible, les points de Tchebychev donnent une bien meilleure apporximation de $f(x)$ que les points équi-répartis. Lorsque N grandi cependant la différence semble s'atténuer quelque peu, du moins pour cette fonction. Prenons $N = 14$ par exemple :



Cette fois les 2 polynômes d'interpolation se ressemble beaucoup, et ont une allure très proche de celle de $f(x)$. L'impact du choix des points, au moins sur cette fonction, semble moindre lorsque le nombre de points augmente.

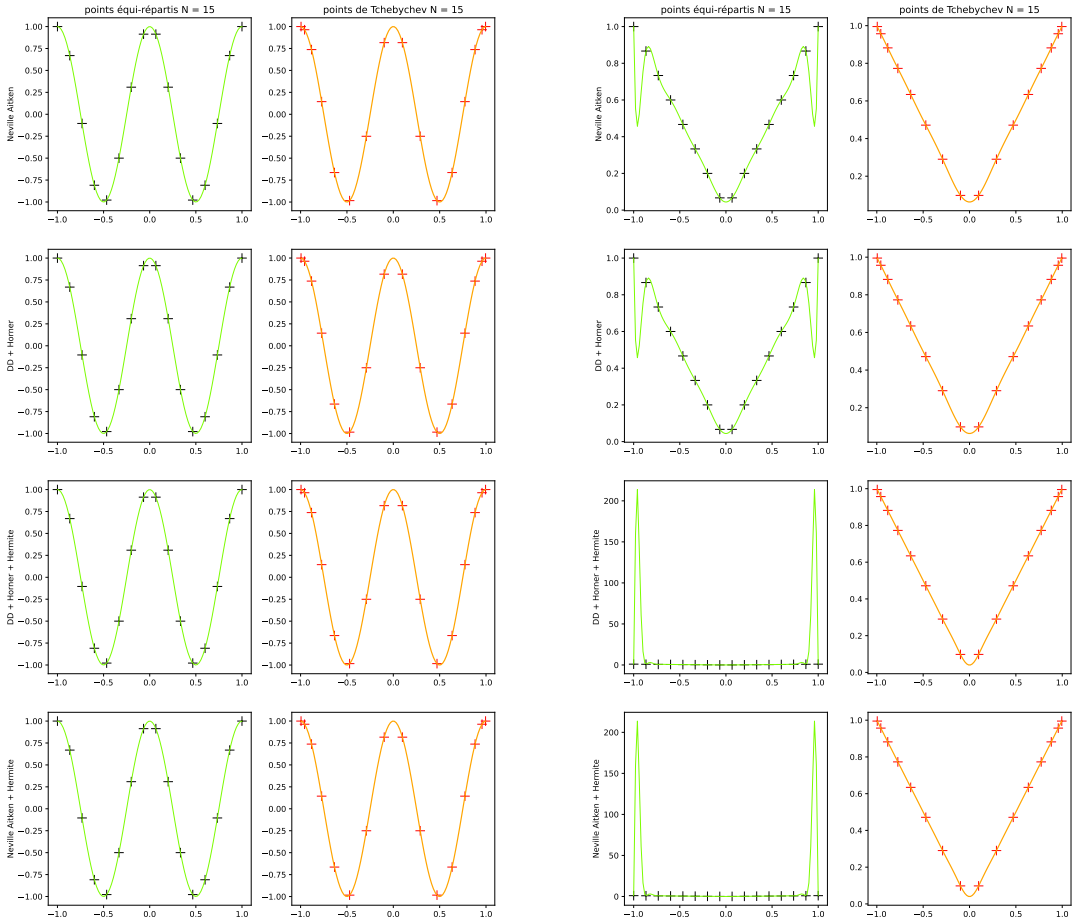
2.2.2. Étude de l'influence du nombre de points d'interpolation

Dans cette partie nous allons nous intérresser à l'impact du nombre de points sur le polynôme d'interpolation. Intuitivement, on pense que plus le nombre de points d'interpolation va être élevé, meilleure sera l'approximation de la fonction par le polynôme d'interpolation. C'est d'ailleurs ce quise produisait dans l'exemple donné juste au dessus. Malheureusement, ce n'est qu'en partie vrai, car dans certains cas, en fonction de la méthode d'interpolation utilisée et de la fonction approximée, l'augmentation du nombre N de points ne permet pas une meilleure approximation de la fonction. On a notamment un célèbre exemple qui est dû à Runge que nous verrons dans la suite. Voyons donc ce qu'il en est dans notre cas. Pour $N = 4$, on a :



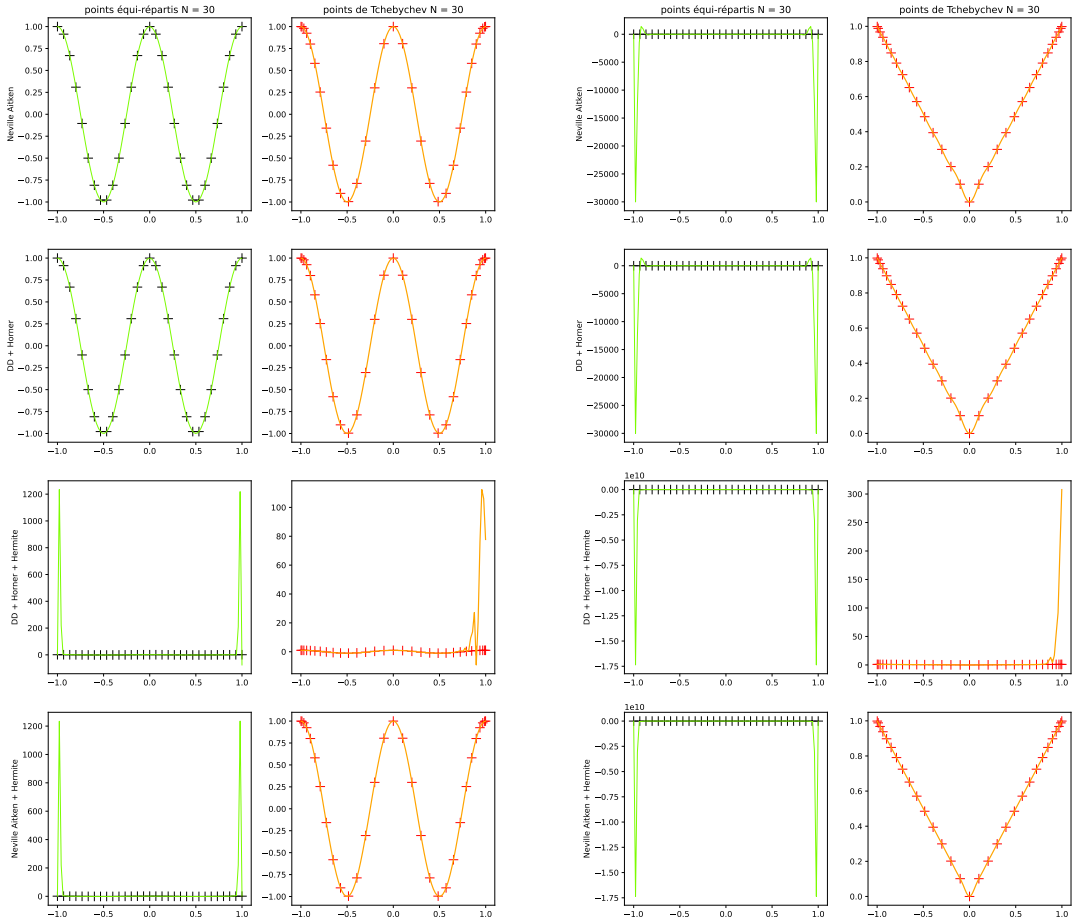
Interpolation des fonctions cosinus et valeur absolue pour $N = 4$

Comme expliquer dans la partie précédente l'interpolation n'est pas de très bonne qualité, bien que cela dépend aussi des méthodes choisies. On obtient cependant un resultat qui se rapproche déjà assez bien des fonctions pour les méthodes d'interpolation d'Hermite pour les deux types de points. Augmentons le nombre de points, pour $N = 15$ on a :



Interpolation des fonctions cosinus et valeur absolue pour $N = 15$

On constate dans ce cas une très nette amélioration en terme de précision d'approximation de f, et ce pour toutes les méthodes pour la fonction cosinus. En revanche, pour la fonction valeur absolue, le résultats est totalement éroné pour les points équi-répartis. On serait tenté de dire que l'augmentation du nombre de points, au moins pour les points de Tchebychev, va être bénéfique. Doublons alors le nombre de points, on obtient ainsi :



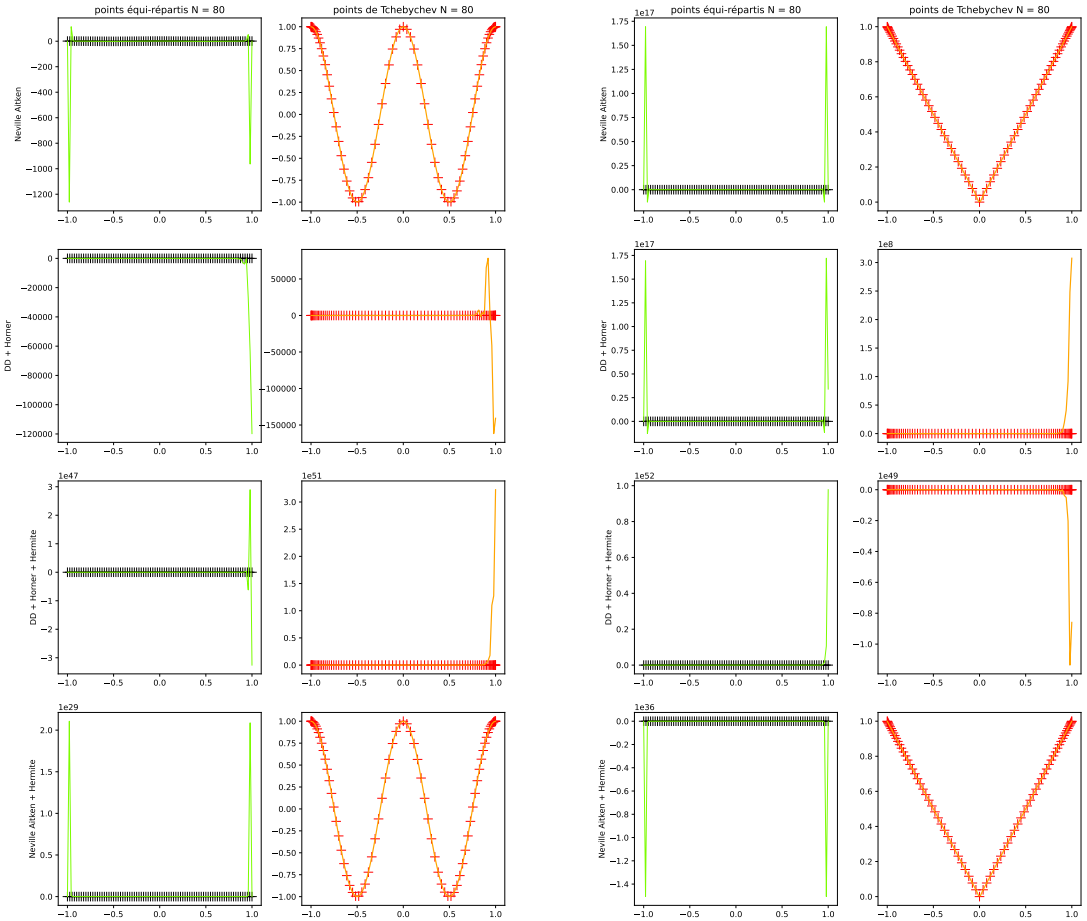
Interpolation des fonctions cosinus et valeur absolue pour $N = 30$

Dans ce cas avec $N = 30$, trois de nos 8 graphiques donnent un résultat complétement abberant pour la fonction cosinus. Précisément ce sont les 2 algorithmes utilisant Hermite qui, ici ne fonctionnent plus. Ici la cause n'est pas le phénomène de Runge, car la fonction approché est un cosinus. La cause de ce dysfonctionnement semble donc provenir de l'augmentation du nombre de points. En effet, l'une des causes probables est que l'on interpole la fonction sur l'intervalle $[-1; 1]$. Ici on prend $N = 30$, soit 31 points, mais globalement plus N augmente, plus les points d'interpolation vont être proches, et leurs différences faibles. De plus on sait qu'en machine étant codé sur un nombre fini de

bits, la plupart des nombres réels ne peuvent pas être représentés de manière exact. Sur 64 bits on dispose d'une précision machine d'environ 10^{-16}. Ainsi en augmentant N, on augmente d'une part le nombre de calculs qui doit réalisé par les algorithmes et d'une autre part on rapproche les points les uns des autres. Or que ce soit l'algorithme de Neville-Aitkens, ou celui des différences divisées+Hörner, on fait à un moment donné une division d'un terme par la différence de 2 points d'interpolations qui sont proches car N est grand. Donc le dénominateur devient petit, et il y a possiblement des propagations d'erreurs, et le résultat peut rapidement exploser si le dénominateur se rapproche de 0. En fait quand N devient grand cela fait ressortir l'instabilité des algorithmes, car demande plus de calculs ce qui propage davantages d'erreurs. D'ailleurs les algorithmes ayant échoué sont ceux utilisant l'interpolation d'Hermite, ce qui n'est pas étonnant car ils demandent de dédoubler les points, on a donc 2 fois plus de calculs, donc plus de possiblités de propagations d'erreurs. C'est en cela que, dans notre cas, l'augmentation trop importante du nombre de points peut-être néfaste.

Pour ce qui est de la fonction valeur absolue seul les points de Tchebychev fonctionnent, ce qui confirme encore qu'ils sont bien meilleure que les points équi-répartis. Cependant ici l'augmentation du nombre de points n'est pas non plus bénéfique car la méthode des différences divisées appliquées à l'interpolation d'Hermite ne fonctionne plus même avec les points de Tchebychev.

Observons maintenant le résultat pour $N = 80$.



Interpolation des fonctions cosinus et valeur absolue pour $N = 80$

Sans surprise le résultat est une nouvelle fois abberant, ce qui confirme bien qu'augmenter le nombre de points au-delà d'un certain rang n'apportera pas plus de précision pout les points équi-répartis. En revanche, on peut noter que l'algorithme de Neville-Aitken s'en sort très bien avec les points de Tchebychev. De plus, la fonction valeur absolue n'étant pas différentiable en 0, il est difficile d'estimer une majoration de l'erreur. L'erreur reste cependant très faible entre la fonction et le polynôme d'interpolation de Lagrange, du moins visuellement parlant.

2.2.3. Étude de l'influence de la méthode d'évaluation utilisée

Intérressons nous maintenant aux différentes méthodes. On va considérer ici 4 méthodes, qui sont Neville-Aitkens et Différences Divisées+Hörner, puis ces 2 méthodes adaptées à l'interpolation d'Hermite au premier ordre.

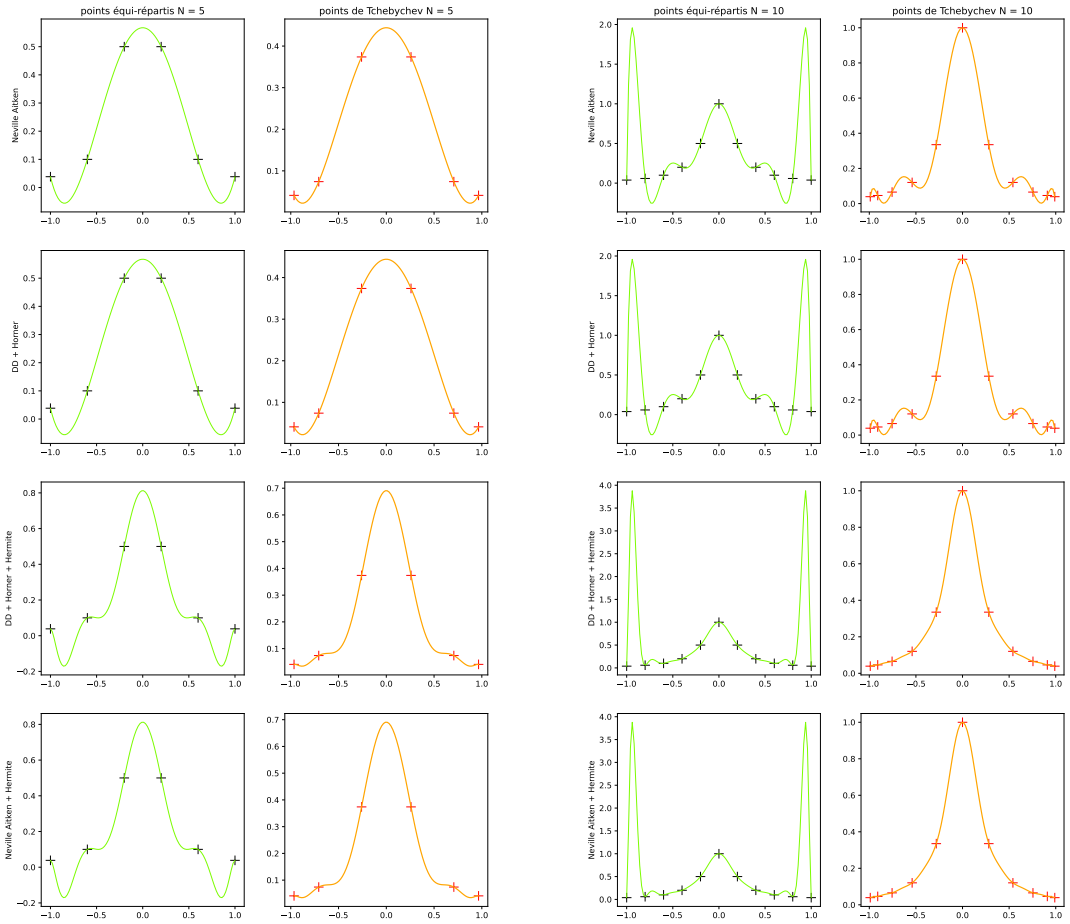
D'après les graphiques des sections précédentes, on voit clairement que lorsque N est petit, par exemple pour $N = 4$, l'interpolation d'Hermite est bien plus précise que les méthode appliquées de manière classique. Ce qui est normal car on impose plus de conditions. Ensuite on a vu précédemment que lorsque l'on augmente N, ce sont d'abord les méthodes utilisant l'interpolation d'Hermite qui donnent des résultats érronés. Cela est sûrement du au fait qu'elles demandent plus de calculs, donc il y a probablement davantage de propagation d'erreurs ce qui conduit à de tels résultats.

Ensuite si on compare Neville-Aitkens et les Différences Divisées, il semblerait que l'algorithme effectuant les Différences Divisées et Hörner soit moins stable que Neville-Aitkens. En effet, si l'on regarde le graphique précédent par exemple, pour $N = 80$, les 2 seuls algorithmes ayant fonctionné sont ceux basés sur Neville-Aitkens. De même pour $N = 30$, l'interpolation d'Hermite basé sur Neville-Aitkens fonctionnait un peu mieux que celle basé sur les Différences Divisées et Hörner. Cependant ce dernier algorithme à l'avantage d'avoir un coût de calcul moindre.

Finalement, notons que lorsque N est petit les algorithmes de Neville-Aitkens et des Différences Divisées+Hörner donnent quasiment les mêmes résultats.

2.2.4. Etude du phénomène Runge

On a vu que l'erreur du polynôme d'interpolation par rapport à la fonction était majorée par $\frac{\|f^{(n+1)}\|_\infty}{(n+1)!}\|v_n\|_\infty$ avec $v_n = \prod_{i=0}^{n}(x - x_i)$.



Comparaison des méthode pour la fonction $f(x) = \frac{1}{1+25x^2}$

Pour des points équi-répartis, lorsqu'on augmente le nombre de points, on constate que le polynôme se met à osciller fortement entre les points x_i avec une amplitude de plus en plus grande, comme l'illustre la figure. On appelle cela le phénomène de Runge, la norme infini de v_n tend vers l'infini lorsque n augmente plus vit que le dénominateur

$(n+1)!$. Les points d'interpolation de Tchebychev permettent de minimiser la $\|v_n\|_\infty$ lorsque le nombre de points augmente, et c'est bien ce que l'on observe sur la figure.

2.2.5. Bonus 1

Nous avons implémenté l'algorithme de Neville-Aitkens pour l'interpolation d'Hermite. Il est accessible à partir du menu s'affichant lors de l'exécution du main.

2.2.6. Bonus 2

Nous avons implémenté les algorithmes de Neville-Aitkens et des différences divisées pour l'interpolation d'Hermite à l'ordre 2. Ils sont accessible à partir du menu s'affichant lors de l'exécution du main.

Part II.

Interpolation de la trajectoire de Moustik

3. Aspect théorique

Pour approcher la dérivée $x_0(t_i)$ (de même pour $y_0(t_i)$), on se propose de considérer le polynôme d'interpolation associé aux points (t_{i-1}, x_{i-1}), (t_i, x_i) *et* (t_{i+1}, x_{i+1}) (resp. $(t_{i-1}, y_{i-1}), (t_i, y_i)$ et (t_{i+1}, y_{i+1})), qu'on notera $S_i(t)$ (resp. $P_i(t)$).

Commençons d'abord par traiter les conditions limites, c'est à dire lorsque $i = 0$, ou $i = n$. Dans ce cas le polynôme d'interpolation passera seulement par 2 points et sera donc une droite. On a donc pour $i = 0$:

$$S_0(t) = \frac{(x_1 - x_0)(t - t_0)}{(t_1 - t_0)} + x_0$$

Et donc en dérivant, on obtient $S_0^{'}$ constante et qui vaut :

$$S_0^{'}(t) = S_0^{'}(t_0) = \frac{(x_1 - x_0)}{(t_1 - t_0)} = x^{'}(t_0)$$

On fait de même lorsque $i = n$, pour obtenir :

$$S_n^{'}(t) = S_n^{'}(t_n) = \frac{(x_n - x_{n-1})}{(t_n - t_{n-1})} = x^{'}(t_n)$$

On procède exactement pareil pour le calcul de $P_0^{'}$ et $P_n^{'}$, ce qui nous donne :

$$\begin{cases} P_0^{'}(t_0) = \frac{(y_1 - y_0)}{(t_1 - t_0)} & si\ i = 0 \\ P_n^{'}(t_n) = \frac{(y_n - y_{n-1})}{(t_n - t_{n-1})} & si\ i = n \end{cases}$$

Exprimons maintenant le polynôme d'interpolation $S_i(t)$ dans le cas général, c'est à dire lorsque $i \in \{1, ..., n-1\}$. On procède par la méthode de Neville-Aitken. Posons :

$$\begin{aligned} t_{i-1} = X_0,\ x_{i-1} = Y_0 \\ t_i = X_1,\ x_i = Y_1 \\ t_{i+1} = X_2,\ x_{i+1} = Y_2 \end{aligned}$$

Remarquons que cette fois nous disposons de trois points, donc le polynôme d'interpolation sera de degré deux.

Étape 0 : On pose :

$$\begin{aligned} T_0^0 = Y_0 \\ T_0^1 = Y_1 \\ T_0^2 = Y_2 \end{aligned}$$

Étape 1 :

$$\begin{aligned} T_1^0(X) &= \frac{T_0^0(X)(X_1 - X) - T_0^1(X_0 - X)}{(X_1 - X_0)} \\ &= \frac{Y_0(X_1 - X) - Y_1(X_0 - X)}{(X_1 - X_0)} \end{aligned}$$

et

$$\begin{aligned} T_1^1(X) &= \frac{T_0^1(X)(X_2 - X) - T_0^2(X_1 - X)}{(X_2 - X_1)} \\ &= \frac{Y_1(X_2 - X) - Y_2(X_1 - X)}{(X_2 - X_1)} \end{aligned}$$

Étape 2 :

$$T_2^0(X) = \frac{T_1^0(X)(X_2 - X) - T_1^1(X_0 - X)}{(X_2 - X_0)}$$

Après développement des calculs et dérivation de $S_i(t) = T_2^0(t)$,on obtient :

$$\begin{aligned} (S_i(t))' = Y_0 &\frac{(2XX_2 - X_2^2 + X_1^2 - 2XX_1)}{(X_2 - X_0)(X_2 - X_1)(X_1 - X_0)} \\ + Y_1 &\frac{X_2^2 - 2XX_2 + 2XX_0 - X_0^2}{(X_2 - X_0)(X_2 - X_1)(X_1 - X_0)} \\ + Y_2 &\frac{X_1^2 - 2X_1X_0 + X_0^2}{(X_2 - X_0)(X_2 - X_1)(X_1 - X_0)} \end{aligned}$$

En évaluant en X_1, on a alors

$$\begin{aligned} (S_i(X_1))' = Y_0 &\frac{(X_1 - X_2)}{(X_2 - X_0)(X_1 - X_0)} \\ + Y_1 &\frac{X_2 - 2X_1 + X_0}{(X_2 - X_1)(X_1 - X_0)} \\ + Y_2 &\frac{X_1 - X_0}{(X_2 - X_0)(X_2 - X_1)} \end{aligned}$$

Finalement,

$$S'(t_i) = \frac{x_{i-1}(t_i - t_{i+1}) + x_i(t_{i+1} - 2t_i + t_{i-1}) + x_{i+1}(t_i - t_{i-1})}{(t_{i+1} - t_{i-1})(t_i - t_{i-1})(t_{i+1} - t_i)} = x^{'}(t_i)$$

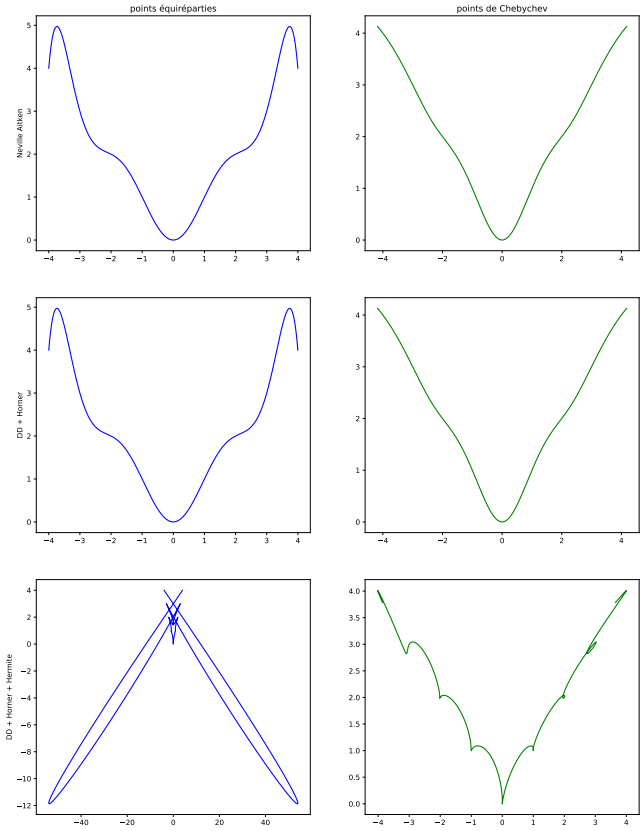
On procède exactement de la même manière pour obtenir $P_i'(t_i)$, donc on l'obtient ici en remplaçant les x_i par y_i.

$$P'(t_i) = \frac{y_{i-1}(t_i - t_{i+1}) + y_i(t_{i+1} - 2t_i + t_{i-1}) + y_{i+1}(t_i - t_{i-1})}{(t_{i+1} - t_{i-1})(t_i - t_{i-1})(t_{i+1} - t_i)} = y^{'}(t_i)$$

4. Code

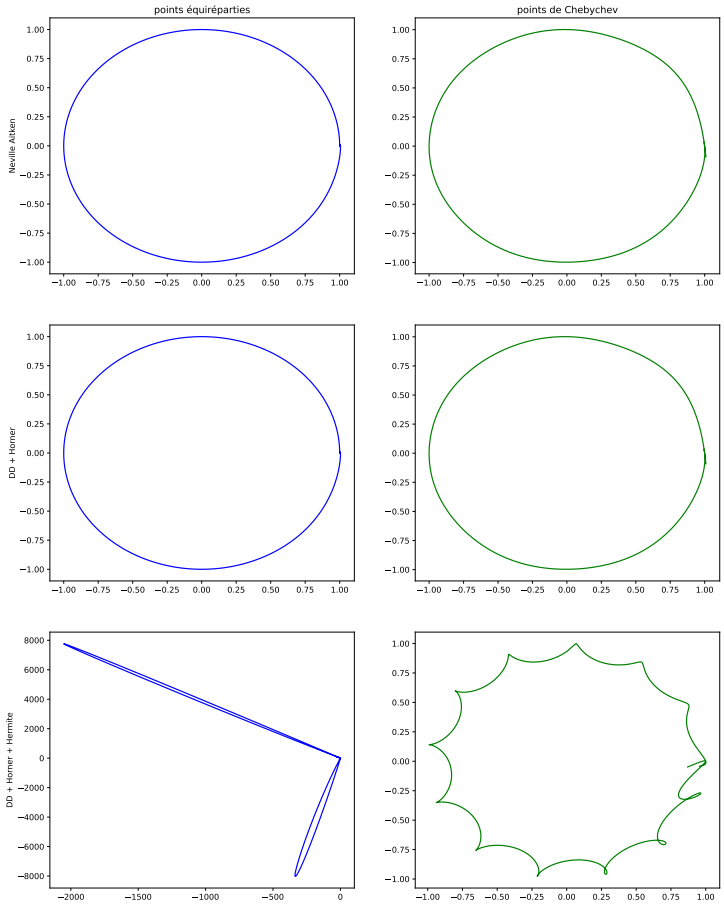
4.1. Test des différentes méthodes sur les jeux de données

Nous allons ici tester les 3 méthodes que sont Neville-Aitkens, les Différences divisées + Hörner, et les Différences divisées adaptées à l'interpolation d'Hermite su différents jeux de données. Commençons par tester ces méthodes sur le jeu de donées dessin3.data, qui est une trajectoire simple. Sur ce jeu de données on obtient :



Les 2 premières méthodes donnent la même chose, cependant on observe une différence importante entre les points équi-répartis à gauche et ceux de tchebychev à droite, qui nous

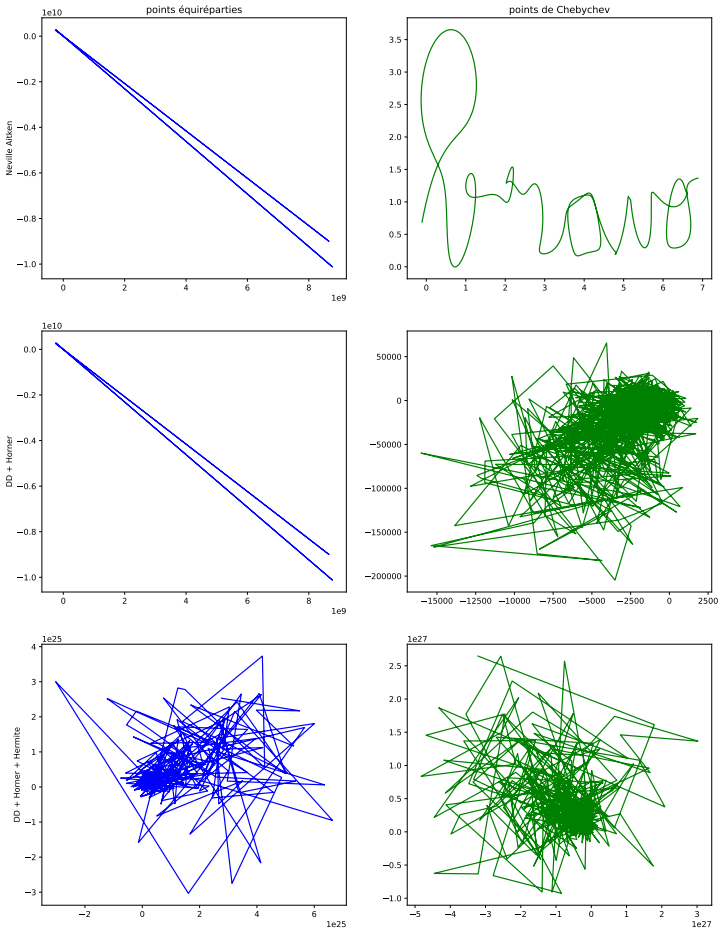
donnent un bien meilleur résultat. La troisième méthode, celle utilisant les différences divisées ne fonctionnent pas. Au vu de l'allure relativement simple de la trajectoire cela est étonnant, mais si il y a une erreur dans notre code ou dans la formule de calcul, nous n'avons pas réussi à la trouver !



En ce qui concerne le cercle, on peut remarquer que les algorithmes de Neville-Aitken et des différences divisées ont l'air de bien fonctionner que ce soit pour les points équirépartis ou bien les points de Tchebychev. En revanche, lorsque l'on passe au problème d'Hermite, les algorithmes nous donnent tous les deux des résultats érronés, encore une fois. On peut quand même noté l'apparence d'un cercle pour les points de Tchebychev.

Pour l'interpolation de la trajectoire de Moustik seul l'algorithme de Neville-Aitkens appliqué aux points de Tchebychev semble fonctionner. Le résultat obtenu est alors le

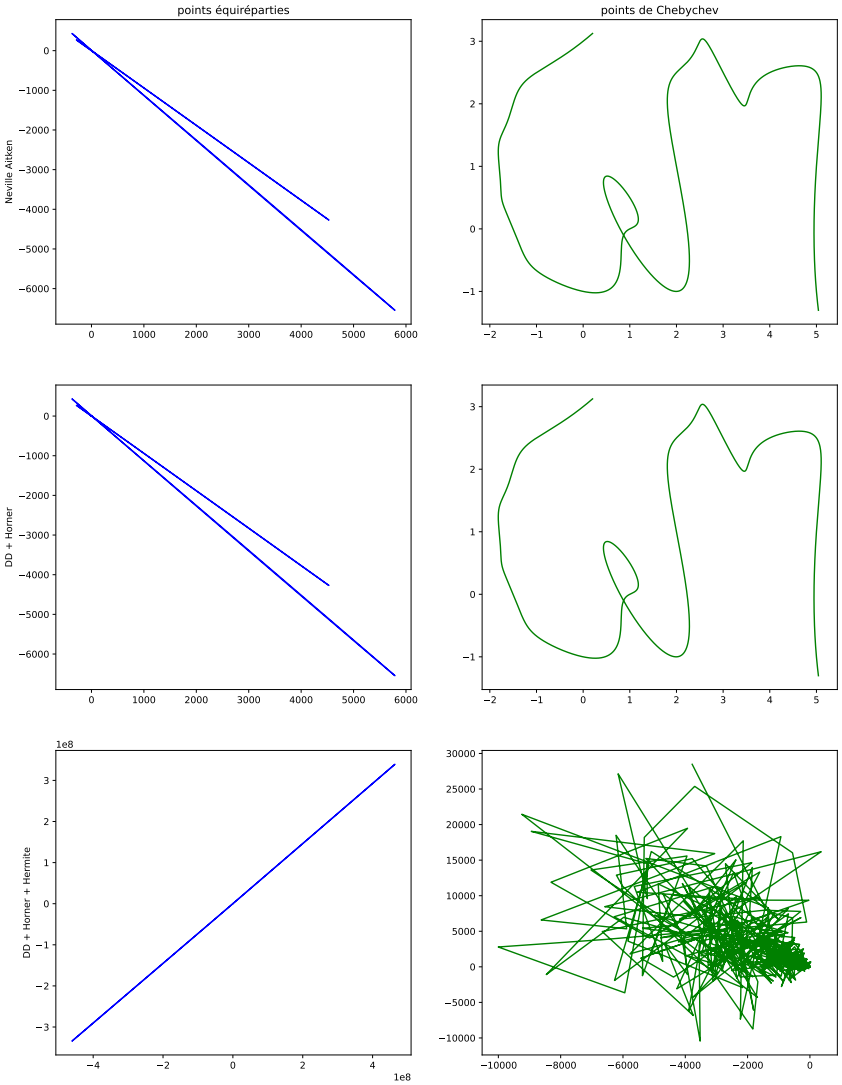
suivant :



Pour toutes les autres méthodes le résultat obtenu est complétement faux.
De plus dans l'algorihtme des différences divisées + Hörner, c'est l'algorithme des différences divisées qui semble être en cause dans le mauvais résultat de l'interpolation et non celui d'Hörner. En effet si à la sortie de l'algorithme des différences divisées on affiche la matrice DD contenant les coefficients de la décomposition dans la base de Newton du polynôme d'interpolation, on s'aperçoit que ceux-ci on des valeurs très grande ce qui laisse penser à des divisions par des valeurs proches de 0 au sein de l'algorithme, donnant lieu à des coefficents abberants. Par ailleurs, l'algorithme d'Hörner ne fait que des additions, soustractions et multiplications, mais pas de divisions. Ces opérations sont donc moins à même de propager d'énormes erreurs, comme peuvent le faire les divisions.

4.2. Nouveau jeu de données (Bonus)

Sous le même format que les jeux de données déjà existant, nous en avons créé un nouveau, dessin4.data, qui donne le dessin suivant :



Conclusion

Pour conclure ce TP nous a permis de mieux comprendre l'interpolation polynômiale, aussi bien sur le plan théorique que sur le plan pratique. Il nous a permis d'observer l'influence du choix des points d'interpolation, de leurs nombre, ainsi que l'influence des différentes méthodes sur le polynôme d'interpolation. Nous avons aussi pu remarquer que ces algorithmes n'étaient pas tous aussi stables numériquement, d'où l'importance de choisir le bon algorithme en fonction du problème traité. Globalement on retiendra que :

- L'algorithme de Neville-Aitkens semble plus satble numériquement que celui des différences divisées associé à Hörner.
- Les points de Tchebychev sont largement meilleurs que les points équi-répartis, et donnent aux algorithmes une meilleure stabilité numérique.
- L'interpolation d'Hermite est meilleure lorsque N est faible, mais les algorithmes l'utilisant semblent un peu plus instables numériquement.
- L'augmentation du nombre de points d'interpolation n'est plus bénéfique à partir d'un certain rang, car cela va induire trop de calculs, et causera des divisons par des nombre très petits ce qui propagera les erreurs et donnera parfois des résultats abberants.