DIF Clément
LOXOL Nicolas

Compte rendu TP : Interpolation polynomiale

1. Rappel des méthodes :

Les méthodes d'interpolation et d'approximation ont pour objectif de retrouver l'équation mathématique d'une courbe traversant une liste de points mesurés.

Dans la suite, N est un entier naturel représentant le nombre de points fournis par la série servant à l'interpolation.

Interpolation de Lagrange :

Soit $P_{n<N}(x) = a_n x^n + a_{n-1}x^{n-1} + \dots + a_1 x + a_o$ un polynôme. Soit $1 \leq i \leq N$; on sait que le point (x_i, y_i) vérifie $P_{n<N}(xi) = yi$.

On veut calculer les coefficients a_n, a_{n-1}, ..., a_0

On écrit le polynôme d'une manière différente :

$$P_{(N-1)}(x) = \sum_{i=1}^{N} y_i l_i(x)$$

Ou l_i est la fonction définie par :

$$l_i(x) = \Pi_{j=1, j\neq i}^{N} \frac{x - x_j}{x_i - x_j}$$

On peut remarquer que $l_i(x_j)$ vaut 0 pour j = i et 1 sinon. La fonction $l_i(x)$ est le produit de N −1 polynômes de degré 1, c'est donc un polynôme de degré N −1.

Interpolation de Newton :

Le polynôme de Newton est aussi un polynôme d'interpolation. Celui-ci peut s'écrire :

$$P_{(N-1)}(x) = b_0 + b_1(x - x_1) + b_2(x - x_1)(x - x_2) + \dots + b_{(N-1)}(x - x_1)(x - x_2)\dots(x - x_{(N-1)})$$

On recherche les coefficients b_0, b_1, ... tels que $P_{N-1}(x_i) = y_i$ avec $1 \leq i \leq N$.

Si on remplace x par x_i, on obtient le système linéaire suivant :

$$\begin{cases} y_1 = P_{(N-1)}(x_1) = b_0 \\ y_2 = P_{(N-1)}(x_2) = b_0 + (x_2 - x_1)b_1 \\ \dots \quad \dots \quad \dots \\ y_N = P_{(N-1)}(x_N) = b_0 + (x_N - x_1)b_1 + \dots + (x_N - x_1)(x_N - x_2)\dots(x_N - x_{(N-1)})b_{(N-1)} \end{cases}$$

On peut d'ailleurs remarquer que la matrice du système est triangulaire inférieure et qu'il peut en conséquence être résolu par substitution connaissant les N points (x_i, y_i). Cependant la résolution par l'utilisation des différences divisées est plus rapide.

On utilise la définition de la différence divisée à l'ordre 1, 2,…, k avec k un entier naturel:

1. La différence divisée de premier degré est $\nabla y_i = \frac{y_i - y_1}{x_i - x_1}, \forall i = 2, \dots, N.$
2. La différence divisée de second degré est $\nabla^2 y_i = \frac{\nabla y_i - \nabla y_2}{x_i - x_2}, \forall i = 3, \dots, N.$
3. La différence divisée de degré k est : $\nabla^k y_i = \frac{\nabla^{(k-1)} y_i - \nabla^{(k-1)} y_k}{x_i - x_k}, i = k+1, \dots, N$

On obtient tous les coefficients du polynôme par les formules suivantes. Celles-ci étant déterminées par récurrence à partir du système défini précédemment.

$$\begin{aligned} b_0 &= y_1, \\ b_1 &= \nabla y_2 \\ b_2 &= \nabla^2 y_3. \\ &\dots \\ b_i &= \nabla^i y_{(i+1)}, i = 3, \dots, N-1 \end{aligned}$$

Le polynôme P_{N-1} est alors obtenu de la manière suivante :

$$\begin{aligned} P_0(x) &= b_{N-1} \\ P_1(x) &= b_{N-2} + (x - x_{N-1})P_0(x) \\ P_2(x) &= b_{N-3} + (x - x_{N-2})P_1(x) \\ &\dots\dots \\ P_{N-1}(x) &= b_0 + (x - x_1)P_{N-2}(x) \end{aligned}$$

Remarques :

- Dans ce TP nous traiterons les fonctions polynomiales comme unique cas.
- Le théorème d'unisolvance (« Etant donnés un ensemble N de points (x_i,y_i), il existe un unique polynôme P_{N-1} de degré maximum (N-1) qui passe par les N points ») répond à la question de l'unicité du polynôme.
- D'autres méthodes plus complexes réalisent l'interpolation. En effet, l'on a parfois recours à des découpages de l'ensemble de points en sous-ensembles que l'on interpole. De plus, l'on s'arrange également pour que les fonctions obtenues se « touchent » sur l'ensemble de points.

2. Présentation du programme :

Notre programme est constitué de plusieurs parties distinctes :

- Les « includes » afin d'utiliser les fonctions de la bibliothèque standard.
- Une structure et un typedef qui nous servira à comparer l'efficacité des deux méthodes d'interpolation.
- Les prototypes qui sont séparés en trois catégories distinctes ; la fonction principale, les fonctions d'interpolations (Lagrange et Newton) et des fonctions annexes nécessaires au bon fonctionnement du programme.
- La fonction main dans laquelle sont injectés nos jeux d'essais et les fonctions définies dans les prototypes.
- Les définitions des fonctions présentées dans les prototypes :
    - *demarrer_methode* → C'est la fonction principale de notre programme, elle permet de choisir la méthode d'interpolation (*int choix* sachant que 1 correspond à Lagrange et 2 à Newton). Les paramètres *double borne_inf* (borne minimale sur l'axe des abscisses) et *double borne_sup* (borne maximale sur l'axe des abscisses) servant à représenter graphiquement le polynôme obtenu.
    - *lagrange* → Cette fonction réalise l'interpolation de Lagrange ; elle prend en paramètre des couples (x_i, y_i) représentés par *double *x* et *double *y* mais également le nombre des dits couples (*int n*). En outre, cette fonction prend également un *double *a* et *double *pa* qui représentent respectivement un ensemble de réels et leurs images par le polynôme trouvé grâce à la fonction. Enfin, la fonction newton prend évidemment le nombre de réels *a* sous forme d'un *int n_a* mais également un pointeur *opt *mesure* permettant de traiter par la suite, les données de temps d'exécution et d'octets alloués.
    - *newton* → Fonctionnement très similaire à la fonction *lagrange.*
    - *calcule_differences_divisees* → Renvoie le tableau contenant toutes les différences divisées pour la fonction *newton*.
    - *afficher_differences_divisees* → Affiche le tableau des différences divisées.
    - *liberer_differences_divisees* → Libération de la mémoire allouée par le tableau des différences divisées.
    - *generer_tab_a* → Alloue un tableau de réels compris entre *double borne_inf* et *double borne_sup* pour la représentation graphique du polynôme.
    - *liberer_tab_a* → Libération de la mémoire allouée par le tableau des réels cités ci-dessus.

Remarques :

- La variable *double li* dans la fonction *lagrange* symbolise la fonction mathématique $l_i(x)$ énoncée dans le premier paragraphe.
- A la fin de chaque boucle, on prend soin de remettre *li* à 1 pour éviter toute erreur de calcul.
- Les données de la série S3 du 3.2 du sujet n'apparaissent pas dans les jeux d'essais car le point 8 possède plusieurs images par le polynôme. (Erreur supposée dans l'énoncé).
- Les résultats nécessaires à la représentation graphique (points a et leurs images) sont directement écrits dans le fichier polynome.pol. Ce dernier nous permettra de tracer

le polynôme via un script python nommé « representer_polynome.py ». Le graphe sera d'une précision notable car le pas entre chaque point s'élève seulement à un millième.

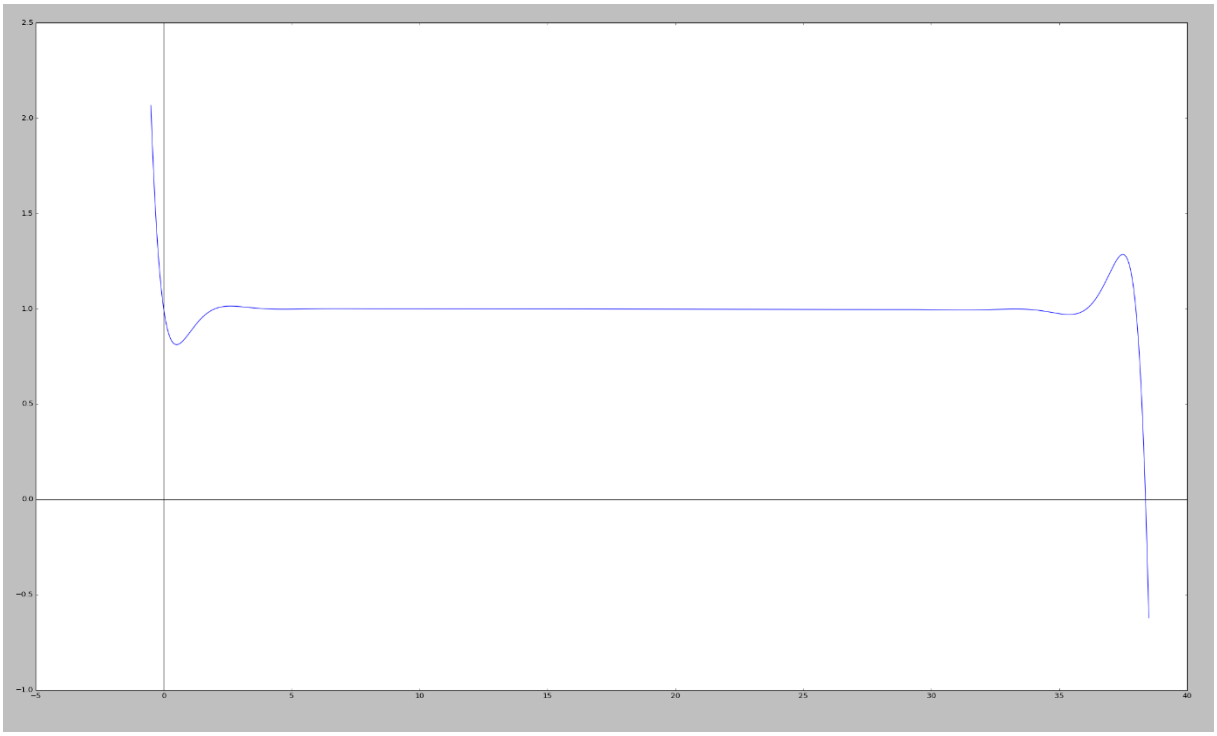
3. Jeux d'essais et graphiques :

Nous utiliserons comme jeux d'essais les tableaux de données présents au verso du sujet ainsi que des tableaux constitués d'un faible nombre de points afin de constater l'oscillation des méthodes lors de l'augmentation du nombre de ces derniers.

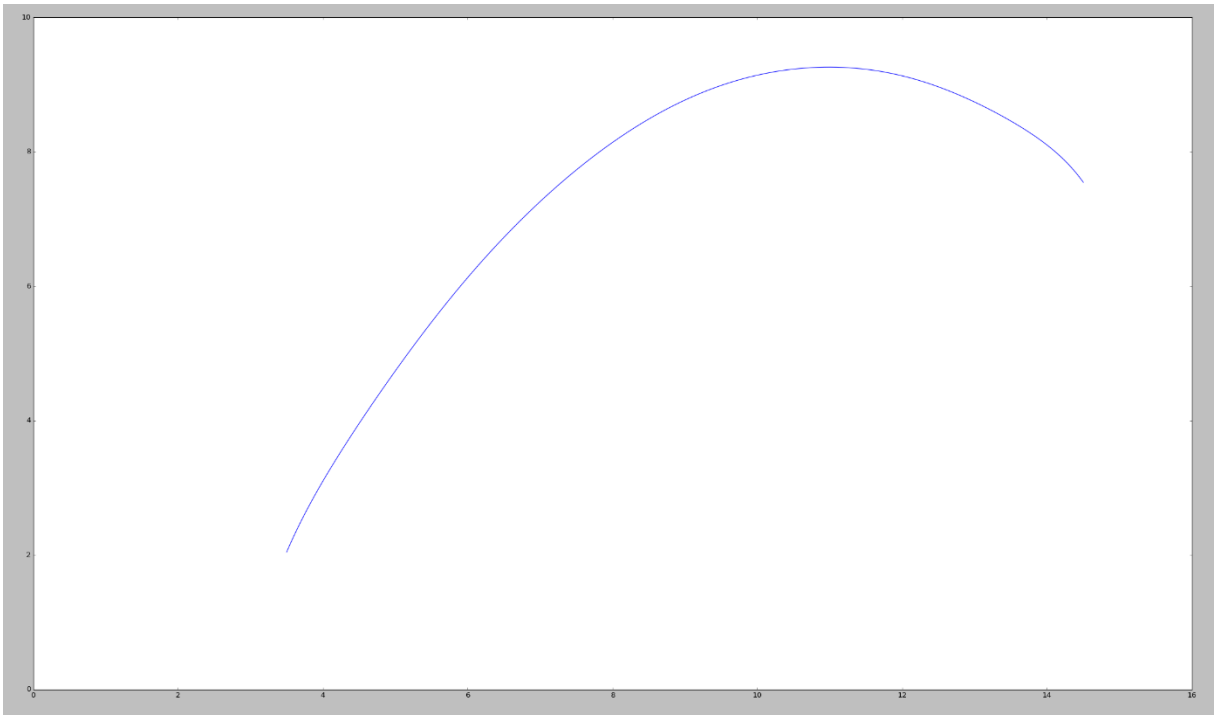
Les graphiques ont été réalisés à partir du langage de programmation Python celui-ci assurant effectivement une meilleure précision qu'un tableur tel qu'Excel.

Il est également à noter que, étant identiques, seul un des graphiques relatifs aux deux méthodes est présent pour chaque série de données. Cela confirme par ailleurs l'unicité du polynôme découvert.

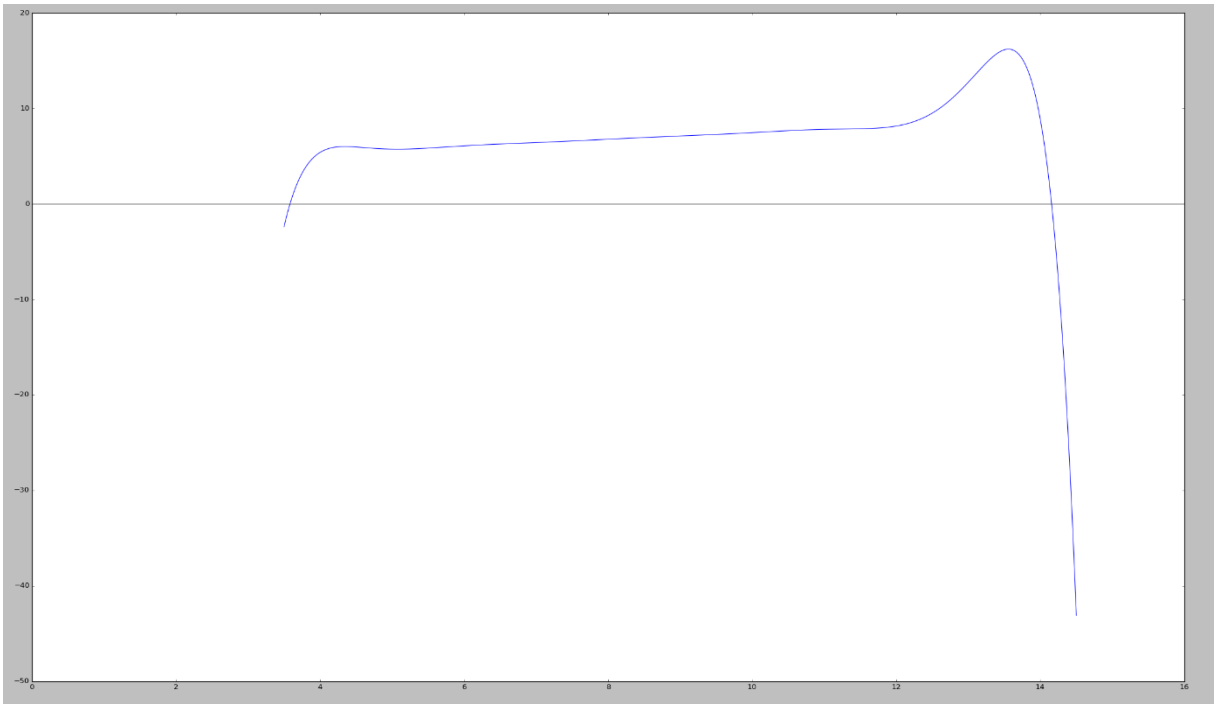
Graphique série sujet 3.1 (20 pts) :



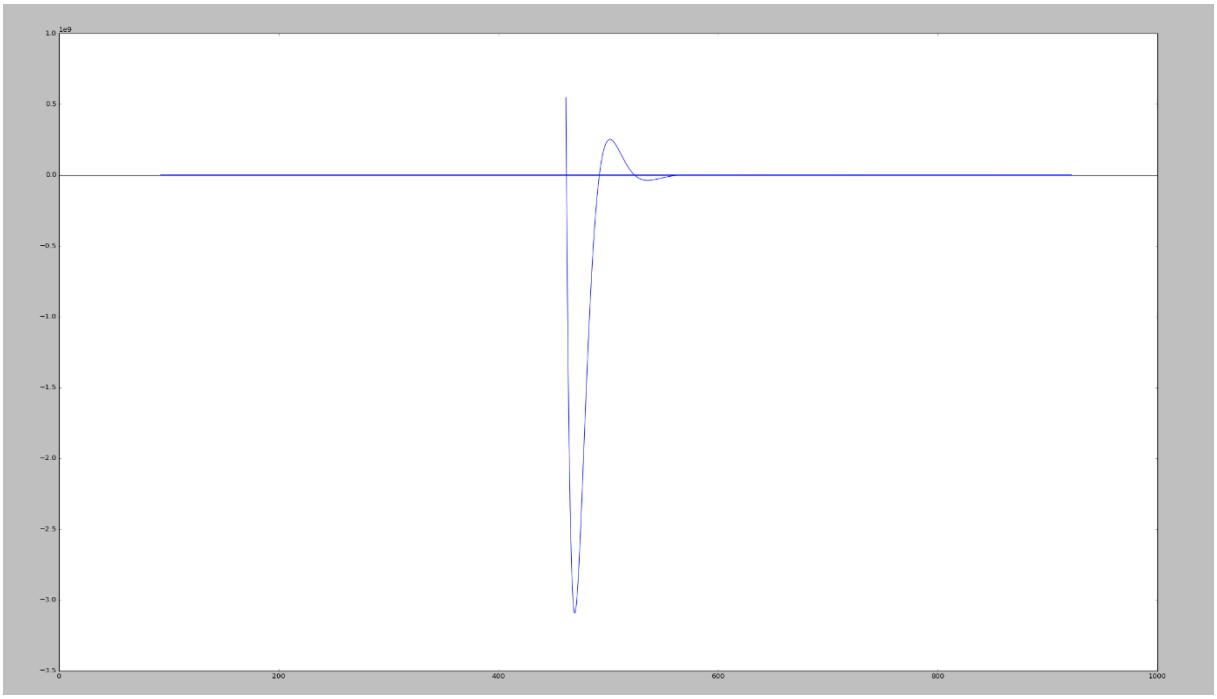
Graphique série sujet 3.2.1 (11 pts) :



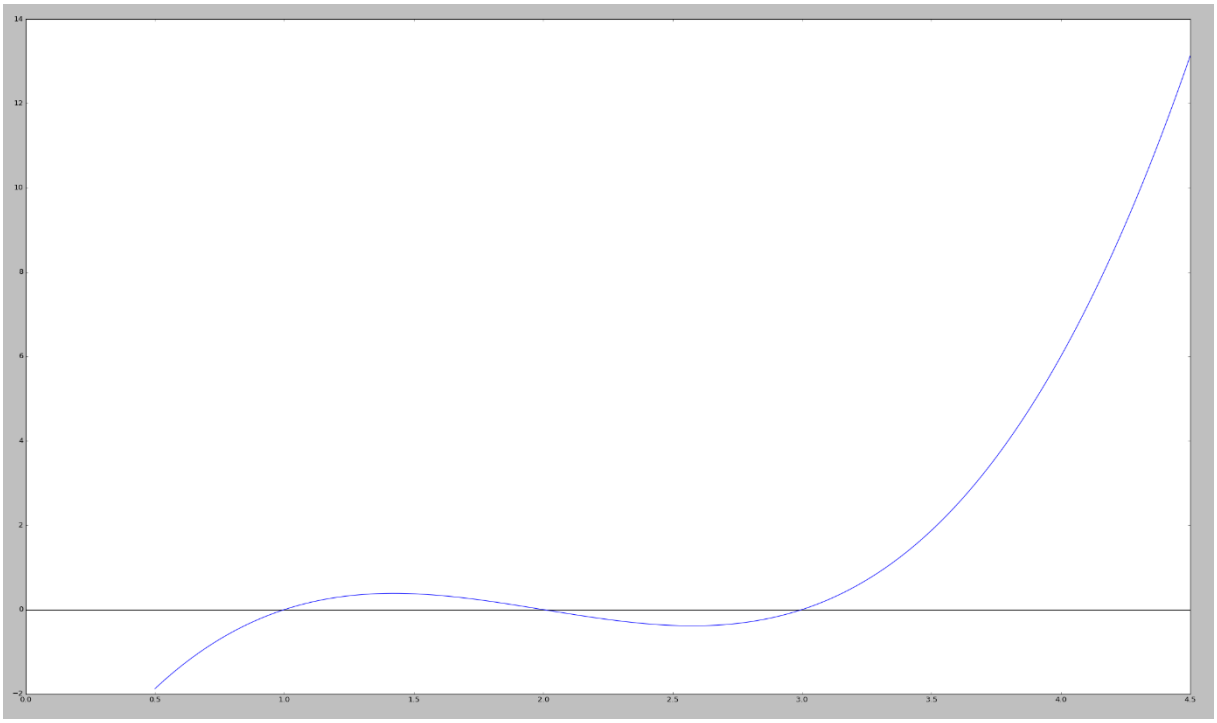
Graphique série sujet 3.2.2 (11 pts) :



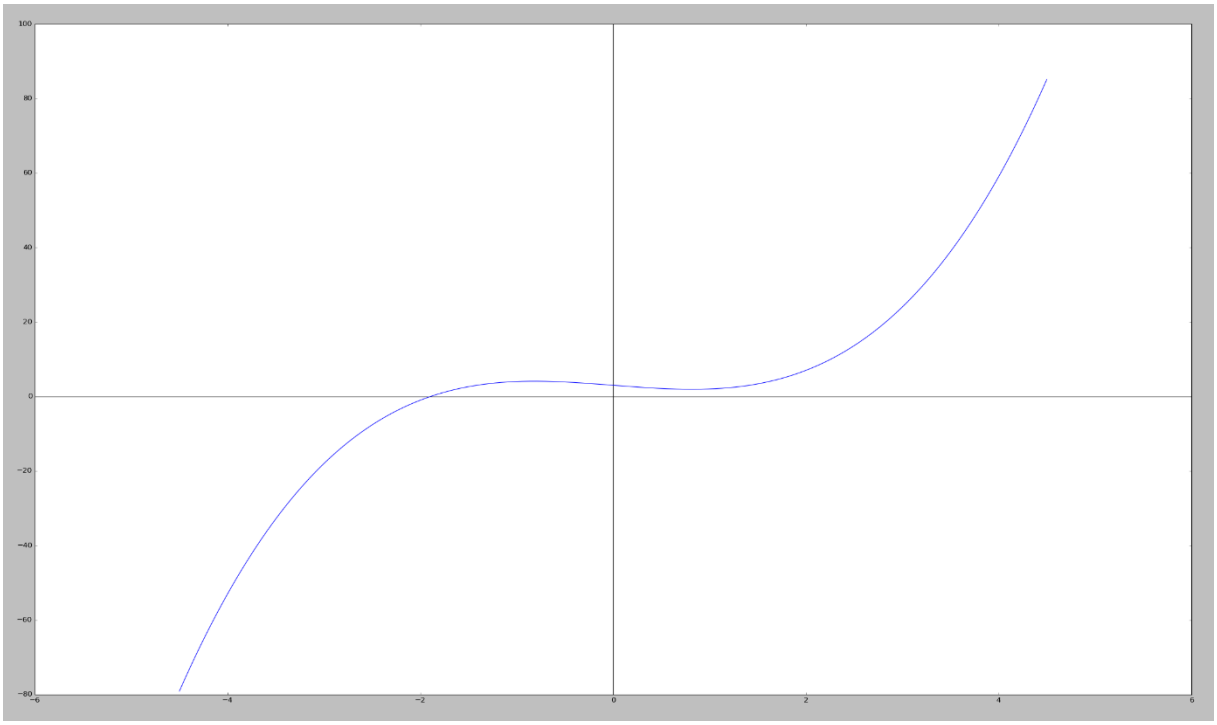
Graphique série sujet 3.3 (21 pts) :



Graphique série personnelle 1 (4 pts) :



Graphique série personnelle 2 (6 pts) :



4. Commentaires des jeux d'essais :

Nombre de points dans la série de données	**Lagrange**		**Newton**	
	Clocks processeur	*Octets alloués*	*Clocks processeur*	*Octets alloués*
4	466	28	179	252
6	2 936	28	816	460
11	11 065	28	3 081	1260
11	10 865	28	3 176	1260
20	130 047	28	32 978	3708
21	1 680 034	28	431 363	4060

Grâce au tableau ci-dessus, on peut noter certains points :

- Pour la méthode de Lagrange lorsque le nombre de points augmente, le temps nécessaire à la résolution augmente de manière exponentielle tandis que le nombre d'octets alloués reste stable. En effet, il n'est pas nécessaire de stocker à un moment

donné des informations contrairement à la méthode de Newton qui nécessite des différences divisées.

- Pour la méthode de Newton lorsque le nombre de points augmente, le temps nécessaire à la résolution augmente également de manière exponentielle. Le nombre d'octets alloués augmente identiquement à cause notamment de l'espace mémoire consommé par les différences divisées comme cité plus haut.
- Nous pouvons donc affirmer que la méthode de Newton est nettement plus rapide que la méthode de Lagrange mais que celle-ci demande à contrario plus de mémoire car elle nécessite l'allocation de tableaux pour la résolution.

5. Conclusion :

La méthode de newton reste la plus efficace pour trouver les interpolés. Cependant il faut tout de même prendre garde à la mémoire que requiert celle-ci. Il peut être en effet plus judicieux de choisir la méthode de Lagrange pour un nombre de points immense pour éviter d'utiliser une quantité trop élevée de mémoire qui pourrait soit amener à un plantage, soit ralentir l'ordinateur.

Nous pouvons également remarquer grâce aux graphiques que lorsque le nombre de points augmente et que, a fortiori, le degré du polynôme augmente, des oscillations apparaissent. Pour éviter cela d'autres méthodes d'interpolation existent…