DIF Clément

LOXOL Nicolas

Compte rendu TP : Interpolation polynomiale

1. Rappel des méthodes :

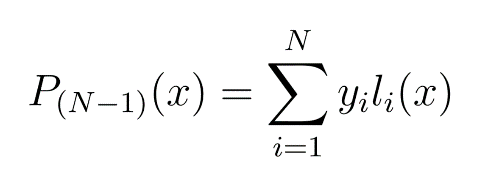
Les méthodes d'interpolation et d'approximation ont pour objectif de retrouver l'équation mathématique d'une courbe traversant une liste de points mesurés. Nous allons vous présenter deux méthodes d’interpolation.

Interpolation de Lagrange :

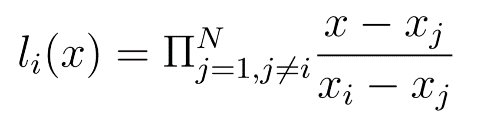
Soit Pn<N (x) = anxn + an-1xn-1 + … + a1x + ao un polynôme. Soit 1<i<N ; on sait que le point (xi,yi) vérifie Pn<N (xi) = yi .

On veut calculer les coefficient an, an-1, ..., a0

On écrit le polynôme d’une manière différente :



Ou li est la fonction définie par :



On peut remarquer que li(xj) vaut 0 pour j 6= i et 1 sinon. La fonction li(x) est le produit de N −1 polynômes de degré 1, c'est donc un polynôme de degré N −1

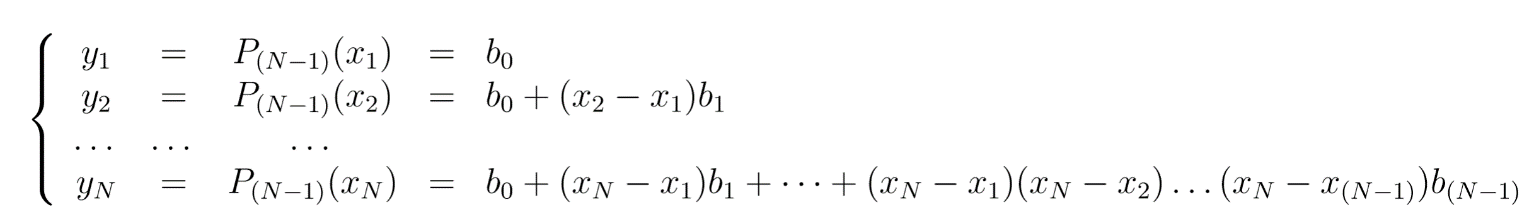
Interpolation de Newton :

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



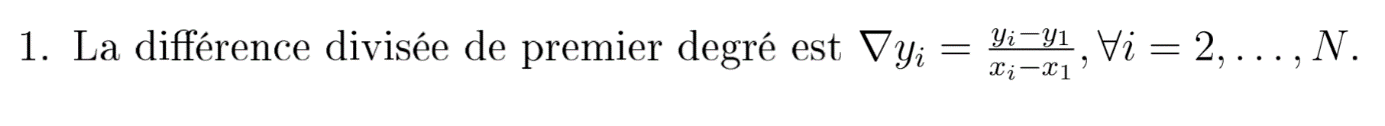
On recherche les coefficients bo,b1,.. tels que PN-1 (xi) = yi avec 1<i<N.

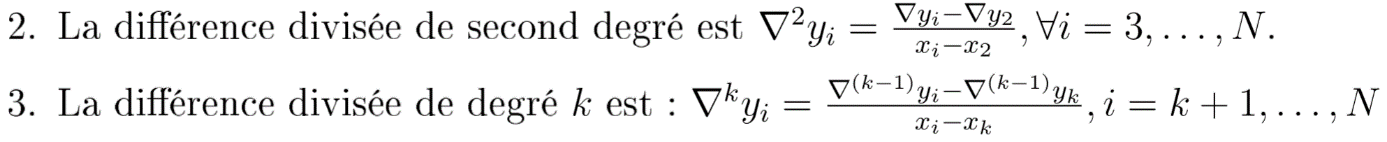
Si on remplace x par xi, on obtient le système linéaire suivant :



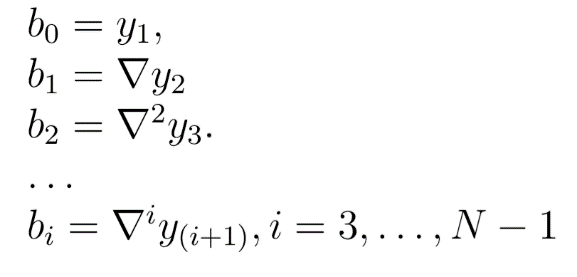
On peut remarquer que la matrice du système est triangulaire inférieure et qu'il peut être résolu par substitution connaissant les N points (xi,yi). Cependant la résolution par l'utilisation des différences divisées est plus rapide.

On utilise la définition de la différence divisée à l’ordre 1,2,…,k :

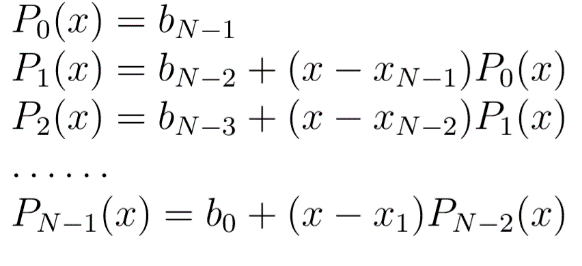




On obtient tous les coefficients du polynôme par les formules suivantes. Celle-ci étant déterminées par récurrence à partir du système défini précédemment.



Le polynôme PN-1 est alors obtenu de la manière suivante :



Remarque :

-Dans ce TP nous traiterons les fonctions polynomiales comme unique cas.

-Le théorème d’unisolvance (« Etant donnés un ensemble N de points (xi,yi), il existe un unique polynôme PN-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. On a parfois recours à des découpages de l’ensemble de point en sous ensemble qu’on interpole. De plus on fait en sorte que les fonctions obtenues se touchent sur l’ensemble de point.

1. Présentation du programme :

Notre programme est constitué de plusieurs parties distinctes :

* Les includes afin d’utiliser les fonctions de la bibliothèque standard et de time.h pour les tests.
* Une structure et un typedef qui nous servira pour comparer l’efficacité des deux méthodes
* 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 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 variables *double borne\_inf* (borne minimale sur l’axe des abscisse) et *double borne\_*sup (borne maximale sur l’axe des abscisse) servant à représenter graphiquement le polynôme obtenue*.*
  + *lagrange* 🡪 Cette fonction réalisent l’interpolation de Lagrange ; elle prend en paramètre (xi,yi) respectivement *double \*x* et *double \*y* puis le nombre de point (*int n*) et pour finir le point que l’on souhaite déterminer (*double a*).
  + *newton* 🡪 Même fonctionnement que la fonction *lagrange.*
  + *calcule\_differences\_divisees* 🡪 Renvoi le tableau contenant toutes les différences divisées afin de faciliter les calculs.
  + *afficher\_differences\_divisees* 🡪 Affiche le tableau des différence divisées. (Le .3f représente la précision fournit dans le tableau).
  + *liberer\_differences\_divisees* 🡪 Libération de la mémoire allouée par le tableau des différences divisées.
  + *generer\_tab\_a* 🡪 Renvoi le tableau des bn nécessaire a la méthode a Newton. (Dans notre fonction les an correspondent au bn du compte rendu)
  + *liberer\_tab\_a* 🡪 Libération de la mémoire allouée par le tableau des an.

Remarque :

-La variable *li* symbolise la fonction mathématique li(x) énoncé dans le premier paragraphe.

- A la fin de chaque boucle, on prend soin à remettre *li* à 1 pour éviter toute erreur de calcul.

-Nous retournons *somme* qui est à la fin de notre fonction la valeur prise par la fonction pour le point *a*.

- Les données S3 du 3.2 n’apparaissent pas car un point (x,y) ne possède qu’une unique image. (Erreur dans l’énoncé)

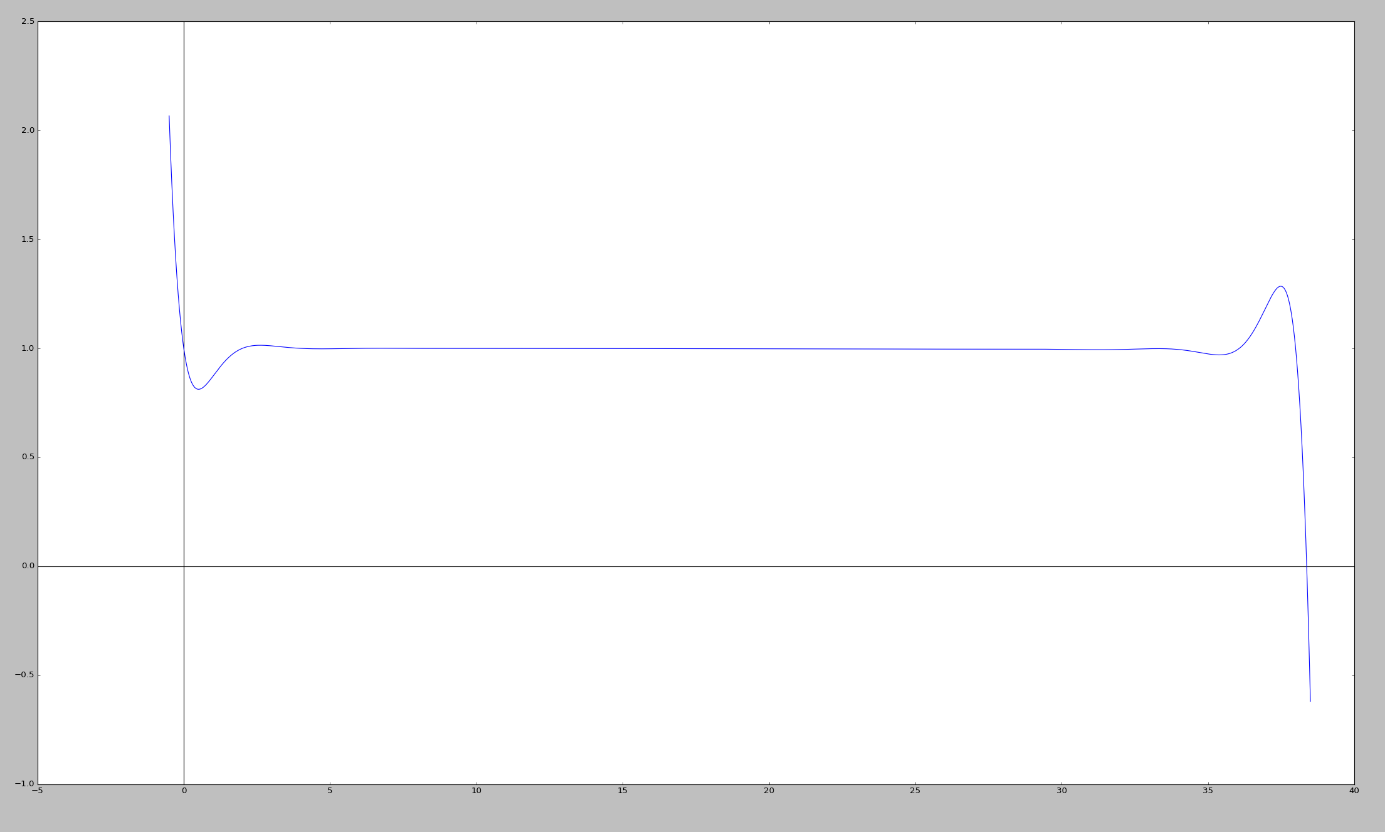
- Les résultats nécessaire (points a et leurs images) sont directement écrit dans le fichier polynome.pol. Ce dernier nous permettra de tracer le polynôme. Le graphique sera extrêmement précis car le pas entre chaque point est de 0.01.

1. Jeux d’essais et graphiques :

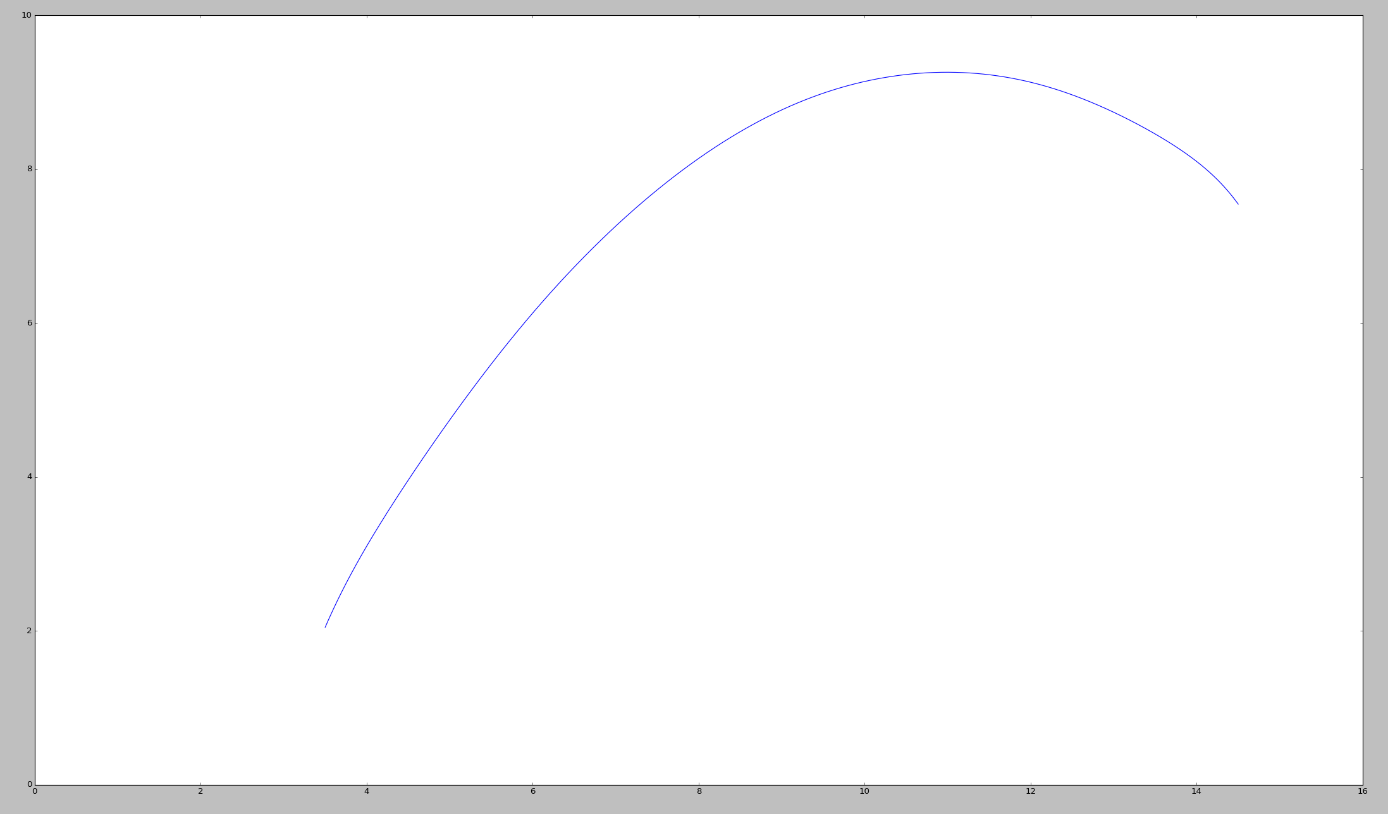
Nous utiliserons comme jeux d’essais les tableaux de donnée présent au verso du TP ainsi que des tableaux constitués d’un faible nombre de point afin de constater l’oscillation des méthodes pour un grand nombre de point.

Graphique : Les graphiques ont été réalisé à partir du langage de programmation python celui-ci assurant une meilleure précision qu’un tableur tel que Excel.

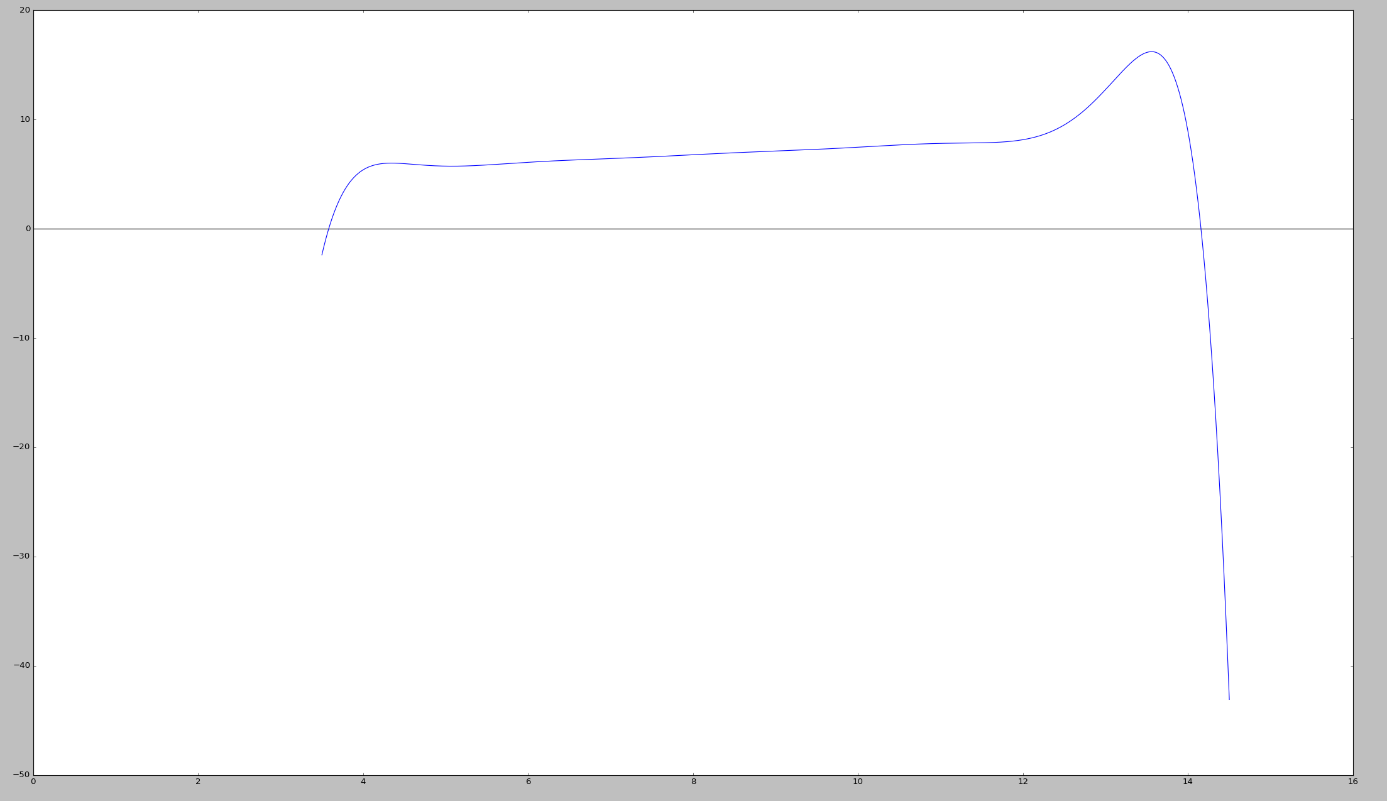
Graphique 3.1 (20 pts) :



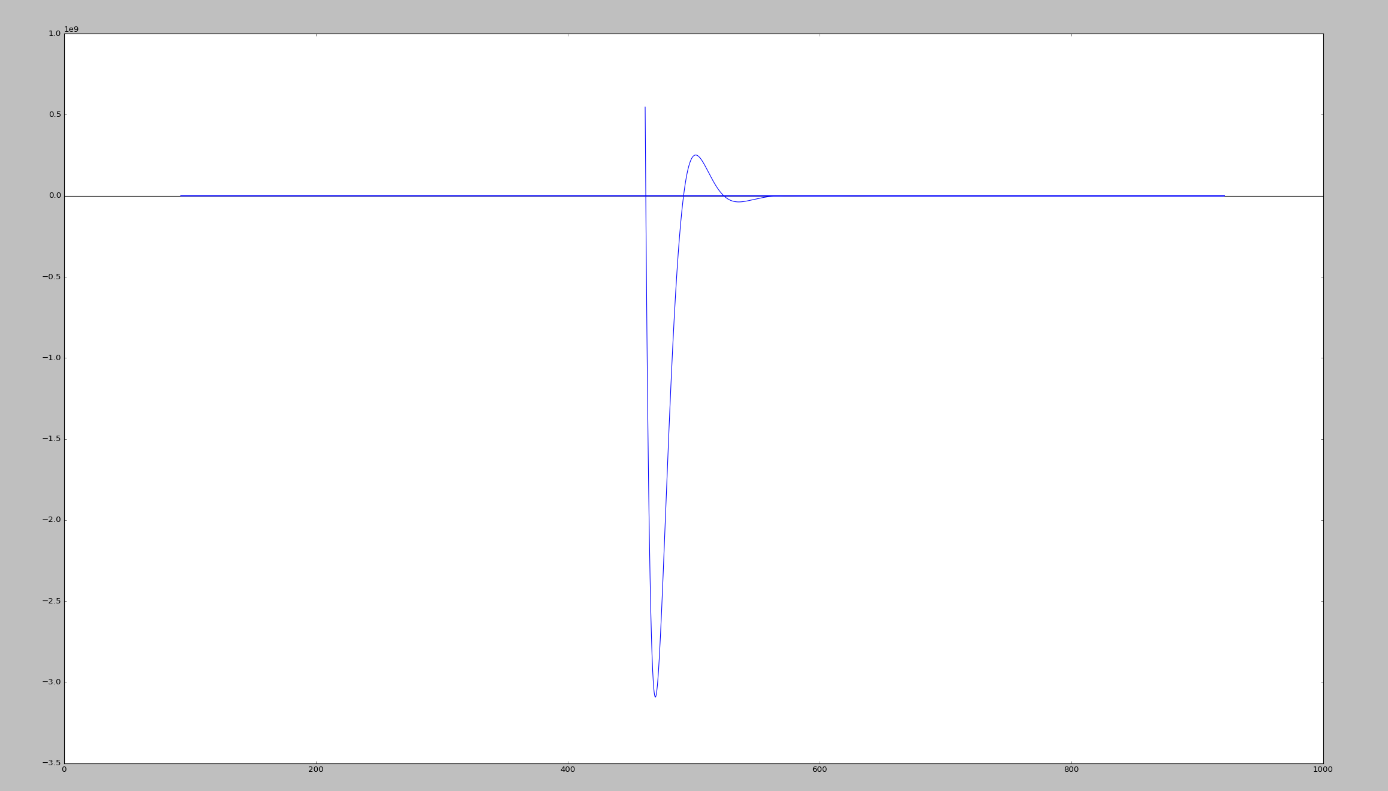
Graphique 3.2.1 (11 pts) :



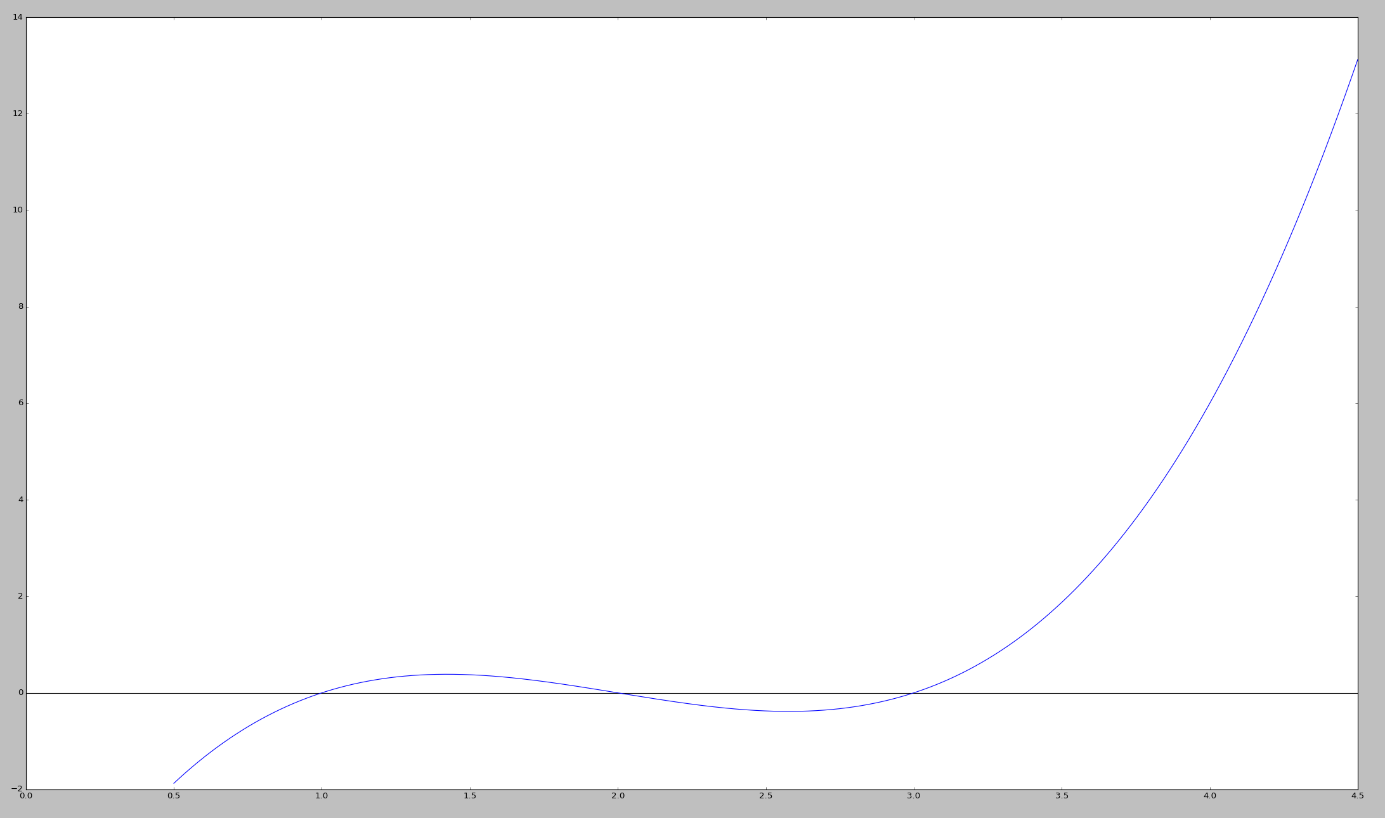
Graphique 3.2.2 (11 pts) :



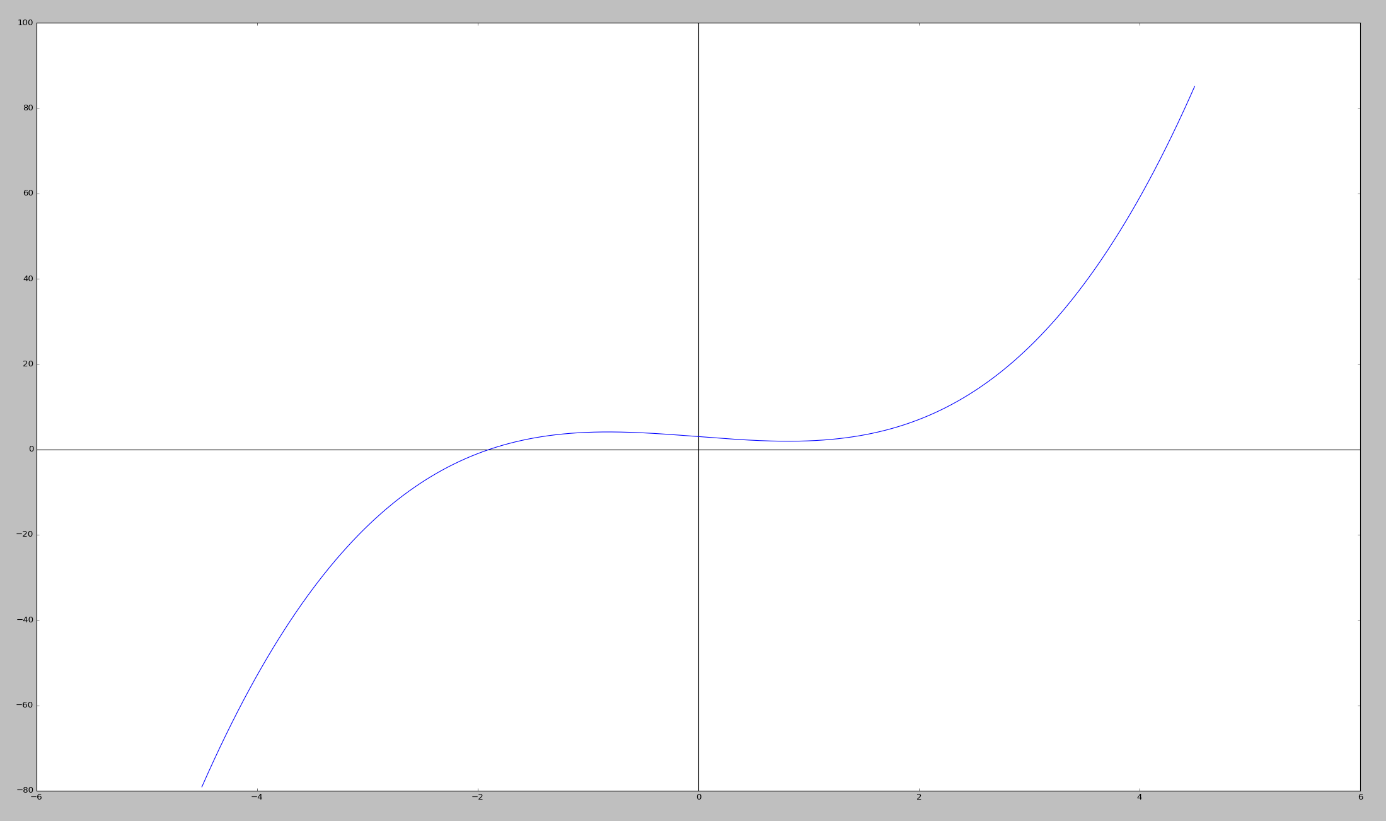
Graphique 3.3 (21 pts) :



Graphique perso\_1 (4 pts) :



Graphique perso\_2 (6 pts) :



1. Commentaires des jeux d’essais :

|  |  |  |
| --- | --- | --- |
| Nombre de point | Lagrange (Clock processeur / Octets alloues) | Newton (Clock processeur / Octets alloues) |
| 4 | 466/28 | 179/252 |
| 6 | 2936/28 | 816/460 |
| 11.1 | 11065/28 | 3081/1260 |
| 11.2 | 10865/28 | 3176/1260 |
| 20 | 130047/28 | 32978/3708 |
| 21 | 1680034/28 | 431363/4060 |

Grace au tableau ci-dessus ; on peut noter certains points :

* Pour la méthode de Lagrange lorsque le nombre de point augmente le temps nécessaire à la résolution augmente de manière exponentielle tandis que le nombre d’octet alloués reste stable.
* Pour la méthode de Newton lorsque le nombre de point augmente le temps nécessaire à la résolution augmente également de manière exponentielle. Le nombre d’octet alloués augmente aussi car la taille des tableaux génères par la fonction augmentent.
* Nous pouvons donc affirmer que la méthode de Newton est nettement plus rapide que la méthode de Lagrange mais que celle-ci demandent plus de mémoire car elle nécessite la construction de tableau pour la résolution.

1. Conclusion :

Il vaut mieux utiliser la méthode de newton pour trouver les interpolé cependant il faut prendre garde a la mémoire que demande cette méthode. Il peut être plus judicieux de choisir la méthode de Lagrange pour un nombre de point immense pour éviter de remplir la mémoire de son ordinateur.

Nous pouvons également remarquer grâce au graphique que lorsque le nombre de point augmente (le degré du polynôme augmente aussi); des oscillations apparaissent. Pour éviter cela d’autre méthodes d’interpolation existent celle-ci procède à un découpage de l’ensemble de point en sous ensemble qu’on interpole. Puis on fait en sorte que les fonctions obtenues se touchent sur l’ensemble de point.