OperaThor

- Baptiste ”Bigfoot” LLORET

(baptiste.lloret)

- Océane ”Sanghgharm” MERLO

(oceane.merlo)

- Sébastien ”Barbie” BARBIER

(sebastien1.barbier)

Date de remise : 22 Février 2018

Table des matières

Introduction

Pour la première soutenance, nous avons tous les trois décider de nous concentrer sur l'essentiel de notre projet, à savoir la résolution d'une équation polynômiale étape par étape. Ces éléments nous tenaient tout particulièrement à coeur car nous voulions au début ne traiter que ces parties et pourquoi pas étudier les polynômes par la suite. Cependant, le projet ayant été jugé trop simple, nous allons par la suite rajouter toutes les parties présentes dans le cahier des charges.

Dans ce rapport de soutenance, nous allons vous montrer ce qui a été fait à ce jour. La première partie de ce rapport est consacré à un rappel du cahier des charges. Nous rappelons dans cette partie comment nous avons formé le groupe, nous y remontrons les tableaux des répartitions des tâches entre nous et au niveau du planning. Enfin, nous rappelons d'où nous est venu l'idée du projet et quels en sont les objectifs.

Dans un second temps, nous nous focalisons sur l'aspect du développement en C de notre projet à ce jour. Nous nous sommes consacrés à la récupération des expressions de polynômes pouvant aller jusqu'au quatrième degré. Ensuite, nous avons traité la résolution des équations du premier degré, du second et enfin du troisième.

Pour clore ce rapport, nous vous parlerons des difficultés que chacun d'entre nous a pu rencontrer en codant. Nous vous souhaitons donc une agréable lecture.

1 Préface

Le but de cette préface est de remettre en évidence des points que nous avons déjà abordés dans le cahier des charges mais qui méritent d'être repris afin de bien se remettre dans le contexte de notre projet. Dans cette préface, nous tenions donc tout d'abord à reparler un peu de notre groupe, de plus, nous voulions rappeler les tâches que nous nous étions fixées afin de comparer le travail prévu avec le travail fourni à ce jour. Enfin, nous vous remettrons en évidence les objectifs de notre projet avant de passer à la phase de développement de ce premier cycle.

1.1 Membres du groupe

Pour nous remettre dans le contexte chronologique de la formation de notre groupe, il faut savoir que nous étions tous les trois destinés à partir en Angleterre dans l'université de Staffordshire. Cela aurait été une excellente expérience pour nous trois et nous aurait permis d'améliorer notre niveau d'anglais. Cependant, nous n'avons pas eu la chance de partir car le partenaire n'a pas respecté les termes du contrat et que nous n'avions pas de passeport pour prendre l'un des plans B que l'on nous avait proposé.

Lors du précédent semestre, nous étions tous les trois dans le même groupe de projet avec un quatrième membre qui lui a pu partir. Il était donc logique que nous nous associons à nouveau car le précédent projet s'était bien passé et nous avons appris le jour même où nous devions rendre les synopsis que nous allions rester à Epita pour ce quatrième semestre.

En lisant la feuille expliquant ce que le projet S4 serait, nous avons vu que les projets alliant programmation et mathématiques auront plus de chance d'être retenu. Suite à cela, nous avons décidé de créer un solveur d'équation étape par étape.

1.2 Rappel des tâches

Pour comparer les tâches faites avec le planning de réalisation de départ, voici les tableaux présents dans le cahier des charges.

Tableau de répartition des tâches principales

	Baptiste	Océane	Sébastien
Reconnaissance d'équation	X		X
Polynôme premier degré	X		X
Polynôme second degré à racines réelles	X	X	
Polynôme de degré trois à racines réelles		X	X
Polynôme de degré quatre à racines réelles		X	X
Equations non polynomiales(approximation)	X	X	
Interpolation polynomiales	X		X
Implémentation de matrice	X	X	
Calculs sur matrices	X		X
Approximation des valeurs propres	X		X
systèmes linéaires et version matricielle		X	X

Tableau de répartition des tâches secondaires

	Baptiste	Océane	Sébastien
Site Web	X	X	
Manuel d'utilisation	X		X
Interface Graphique		X	X

Planning de réalisation

	1ere soutenance	2e soutenance	3e soutenance
Reconnaissance d'équation	I	II	X
Polynôme premier degré	X	X	X
Polynôme second degré à racines réelles	I	X	X
Polynôme de degré trois à racines réelles	I	II	X
Polynôme de degré quatre à racines réelles		II	X
Equations non polynomiales(approximation)		II	X
Interpolation poynomiales		II	X
Implémentation de matrice		II	X
Calculs de matrice	I	II	X
Approximation des valeurs propres			X
Système linéaire et version matricielle	I	II	X
Site Web		II	X
Manuel d'utilisation		I	X
Interface Graphique		II	X

I : Commencé

II : Avancé

X : Terminé

1.3 Objectif du projet

Evidemment notre projet n'est pas une innovation qui n'est pas présente sur le marché et qui révolutionnera l'avenir de la planète ou même d'une génération mais il n'en est pas notre but pour autant. Des systèmes du type existent déjà comme les calculatrices qui donnent directement le résultat voulu ou bien des sites internet comme solumath.com pour n'en citer qu'un trouvé gratuitement sur la toile. Ce site permet de résoudre une équation de type polynôme étape par étape.

Notre projet est développé sous Linux en C afin de correspondre aux exigences demandées. Nous avons pour but de créer un logiciel simple d'utilisation et qui est utile. Notre logiciel est sur le terminal et demande à l'utilisateur de rentrer une équation. Pour cette soutenance, nous ne traitons que le premier, le second et le troisième degré. La résolution des équations se fait étape par étape et est indiquée sur l'écran.

Ce projet sera utile pour toute personne étudiant à l'école de tout niveau et qui voudra vérifier ses calculs après les avoir faits en sachant étape par étape ce qu'il aurait dû faire s'il s'était trompé. Le but de notre logiciel est de pouvoir servir à la fois aux élèves de niveau collège, mais aussi aux élèves de niveau supérieurs comme des élèves de classes préparatoires avec la résolution d'un déterminant de matrice ou bien de la trace d'une matrice par exemple. De plus, une fois implémenté, ce logiciel pourra aider les élèves ayant des difficultés étant donné l'explication ligne par ligne proposée par "OperaThor".

2 Développement

Afin de développer notre projet pour cette première soutenance, nous avons essayé au mieux de respecter les tableaux fixés au cahier des charges même si nous savions que certains éléments seraient plus avancés que prévu et que d'autres le seraient moins. En voyant notre avancé nous sommes assez satisfaits du travail fourni car nous respectons les délais pour la plupart des tâches.

Pour cette première soutenance, nous nous sommes focalisé sur les points importants de notre projet. Comme nous voulions un solveur d'équation étape par étape, ce sont donc les tâches que nous avons décidé de traiter.

Nous avons donc traité la partie reconnaissance d'équation sur le terminal. Nous sommes capables à ce jour de récupérer une équation polynomiale sous une écriture spécifique et de la stocker afin d'y extraire les coefficients se trouvant à gauche et ceux à droite du signe "=".

De plus, Nous avons décidé de résoudre des équations polynômiales de degrés allant de 1 à 3 pour cette soutenance. Lors de ces résolutions, nous affichons étape par étape ce qu'aurai dû faire l'utilisateur s'il avait résolu par lui-même l'équation tappée dans le terminal.

Enfin, afin de ne pas seulement coder les résolutions d'équation qui correspond à la majeure partie de notre projet, nous avons aussi implémenter quelques calculs sur les matrices afin de respecter le cahier des charges.

2.1 Reconnaissance d'équation

Avant même de pouvoir résoudre l'équation voulue par l'utilisateur, la tâche primordiale est de repérer ce que l'utilisateur écrit afin de donner les bons paramètres aux fonctions qui traitent les informations. En effet, lorsque nous écrivons dans le terminal, nous ne rentrons en réalité que des pointers sur des caractères et il n'y a aucune façon pré-faite de dissocier les nombres des lettres. C'est pour cette raison que nous avons dû avoir besoin d'un certain nombre de fonctions.

Afin que le programme puisse résoudre une équation, il faut lui passer l'équation sous la forme

aXˆ2+bXˆ1+cXˆ0=dXˆ2+eXˆ1+f

où a, b, c, d, e, f sont les coefficients des termes des polynômes qu'ils soient situés à gauche ou bien à droite du signe égal. Sachant qu'il est impossible d'obtenir les racines d'un polynôme à l'aide d'une formule pour un polynôme de degré 5 et supérieur selon Abel et Galois, nous allons donc nous limiter dans notre projet aux polynômes de degré 4. Afin de stocker les coefficients, nous avons conclu après maintes réflexions qu'une structure était le meilleur moyen de stocker les informations utiles. La structure est donc de la forme.

```
struct coeff {
    int *left, *right;
};
```

Comme nous pouvons les voir, la structure coeff possède deux pointers *left et *right. chacun de ces pointers sera alloué de 5 emplacements contenant des 0 grâce à *call oc(5, sizeof(int))*. Nous obtenons donc deux tableaux *left et *right de la forme [0 ; 0 ; 0 ; 0 ; 0] où la position dans le tableau est le degré du coefficient. Par exemple :

$$-4X^3 - 2X^2 + 3X - 5$$

écrit bien évidemment sous le format imposé lors de cette première soutenance, sera stocké dans le pointeur *left qui contiendra comme valeurs [-5 ; 3 ; -2 ; -4 ; 0].

Pour mettre les valeurs au bon endroit nous avons besoin d'une fonction d'identification de termes et qui range les coefficients à la bonne place du tableau. Pour ce faire, nous avons une fonction de la forme :

*struct coeff *PutExpression(char *str) ;*

Cette fonction initialise tout d'abord notre structure de coefficient. Ensuite, nous allons lire chaque terme tant que nous ne sommes pas arrivé à la fin de la chaine des caractères donnés en paramètre. Pendant la lecture, nous allons passer les lettres et stocker les chiffres. Les nombres correspondant aux coefficients seront gardés en mémoires dans un pointeur temporaire *char *nb* avant d'être convertis grâce à la fonction *int atoi(char *str) ;* déjà existante mais qui peut être recodé très facilement comme ceci :

```
int my_atoi(char *str) {
        int n = 0 ;
        int d = 1 ;
        while (*str) {
                n = n * d + (*str - '0')
                d *= 10 ;
                str++ ;
        }
        return n ;
}
```

Pour récupérer le bon nombre, nous utilisons la fonction :

*int FindNumber(char *str, char *nb) ;*

Cette fonction va prendre en argument l'indice actuel de notre chaine de caractère et un pointeur sur un espace mémoire où l'on va stocker. Ce nombre est alloué à 42 espaces par habitude de ce nombre sachant qu'il n'y a que peu de chance qu'un utilisateur rentre plus de 42 chiffres. De plus, le type int risque d'overflow bien avant les 42 nombres retenus avec la conversion atoi(). Cette fonction va donc ranger dans la structure coeff les nombres repérés aux indices correspondant aux degrés.

2.2 Equations du premier degré

Nous avons pour cette soutenance, voulut traiter la résolution des équations de degrés inférieur et égal à 3. Pour les équations du premier degré, nous nous sommes limités pour l'instant, à la résolution avec seulement une constante après le égal comme par exemple :

$$4x + 1 = 3.$$

L'utilisateur est guidé et les étapes sont décrites très précisément afin que celui-là puisse comprendre la démarche étape par étape.

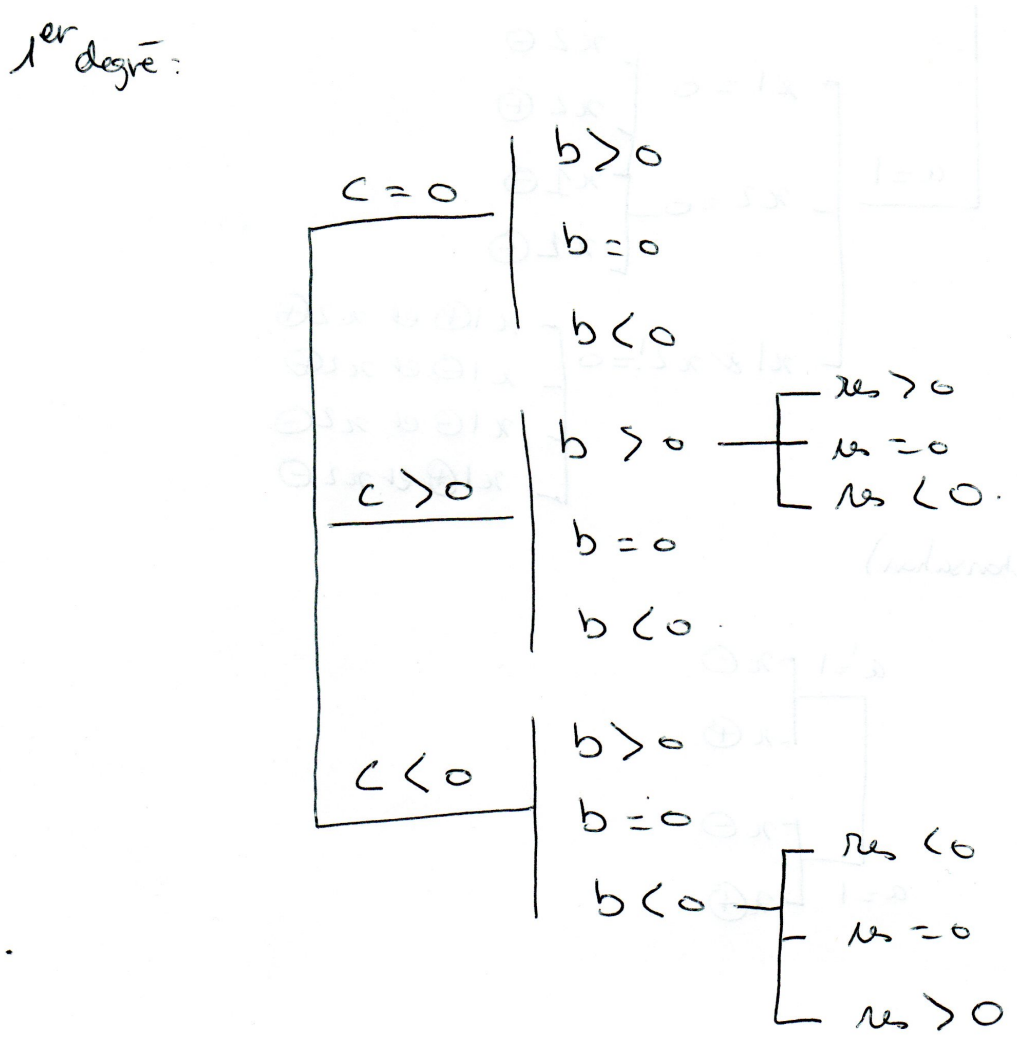
Ici aussi, cette partie a été longue car il y avait beaucoup de signes des paramètres à vérifier. En effet, on a ici le signe de b et de la constante à vérifier.

Donc, dans le schéma fourni ci-après, il y a trois grands cas qui entraînent parfois trois sous petits cas. Les grands cas sont lorsque la constante est positive, nulle ou négative. Si la constante est positive alors, lorsque celle-ci va changer de côté elle sera négative donc il faut mettre un moins à la place du plus et inversement si elle est négative. Pour le cas de si elle est nulle alors ça ne change rien.

Viens ensuite l'addition ou la soustraction de la constante avec le b. Il y a donc les trois sous petits cas (si c et b sont de même signe, il n'y a pas de problème). Si le résultat de l'addition est négatif alors il faut mettre un moins, s'il est positif, alors il faut mettre un plus et s'il est nul, il ne faut pas l'afficher. Cela fait donc plusieurs cas à gérer.

Il vient ensuite l'isolement du x qui n'est pas seulement une manipulation d'affichage. Il se fait dans les trois sous petits cas ou juste après l'addition de b et c s'il n'y a pas de soucis de signes.

Dans le cas où la constante est positive avant le changement de côté, elle sera négative après et inversement si elle est négative. Ensuite il suffit juste de passer la constante appliquée à x de l'autre côté, d'effectuer la division et de l'afficher.



2.3 Equations du second degré

Vient ensuite tout ce qui était traitement d'équations du second degré et de leur affichage. L'utilisateur lorsqu'il demande la résolution d'une équation du second degré, obtient toutes les informations nécéssaires à cette résolution avec une explication détailler et parfois des rappels de cours.

Rappel d'une equation du second degré :

$$aX^2 + bX + c = 0$$

où a, b, c sont des réels

Ainsi, avec cette résolution il obtient pour une équation du second degré, tout d'abord le delta de formule :

$$\Delta = b^2 - 4ac$$

Une petite fonction calcule le delta, et deux fonctions pour les solutions. Le premier renvoi un tableau d'int avec les deux solutions, si le delta est positif et le second renvoi un int pour la solution unique dans le cas du Delta nul. Nous avons utilisé des floats afin d'obtenir des solutions précises et non arrondies, car nous avons déjà arrondi la racine carrée à l'entier près. Pour rappel, les formules pour récupérer les solutions sont pour $\Delta > 0$:

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \qquad et \qquad x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

pour $\Delta = 0$:

$$x_0 = \frac{-b}{2a}$$

et pour $\Delta < 0$: Il n'y a pas de solutions réelles.

Cela n'a pas vraiment posé de problèmes au niveau de l'affichage. Il y a trois cas donc trois affichages différents, l'un pour le delta nul donc une seule solution à afficher, cela n'a pas été trop dur. Le second affichage pour le delta négatif, rien de compliquer non plus. Comme nous nous sommes mis dans l'ensemble réel il n'y a "pas de solutions" (mais dans l'ensemble des imaginaires oui). Le troisième

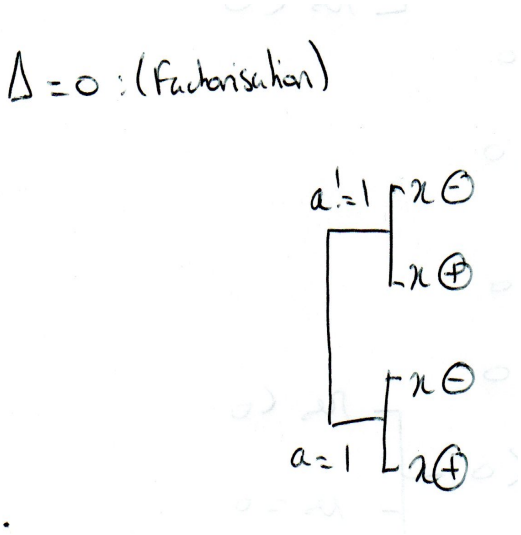
affichage est pour le delta positif donc deux solutions, pas compliqué non plus. mais il y a des sous-cas pour le Delta positif et le Delta nul.

Après la résolution de l'équation et de la recherche des racines grâce au Delta, vient la factorisation du polynôme. Là, ce fut plus long et il a fallu beaucoup de concentration pour n'oublier aucun cas. En effet comme la factorisation d'un polynôme s'écrit de la forme :

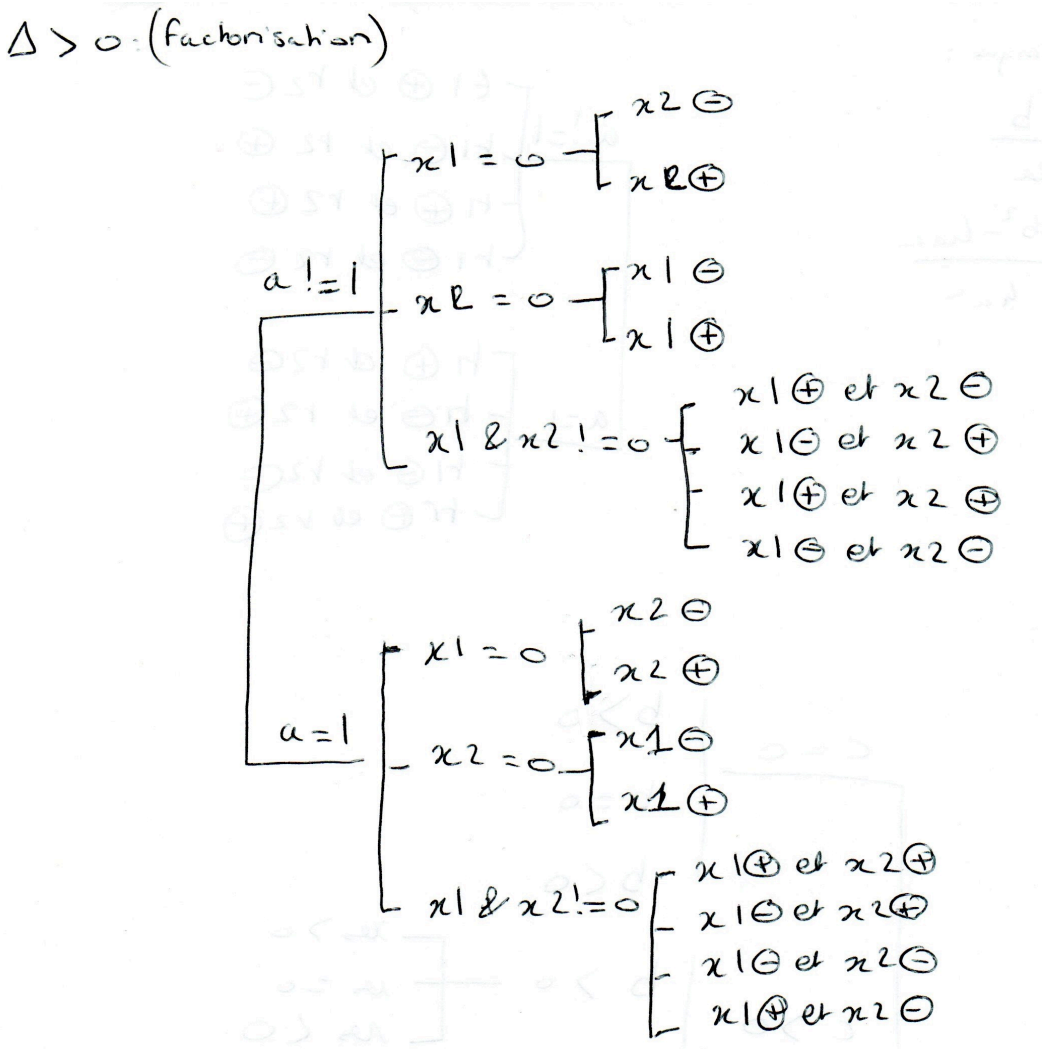
$$a(x - x_1)(x - x_2)$$

Il fallait gérer tous les cas d'affichages. C'est-à-dire, le cas du a, du x_1 et du x_2. Pour le Delta négatif il n'y a pas eu de sous cas à traiter. Cependant, pour ce qui est des deux autres, on ne peut pas en dire autant.

Pour le Delta nul seulement quatre affichages ont été nécessaires. Sur le schéma on peut voir qu'il y a deux grands cas et deux sous-cas pour chaque. Les deux grands cas portent sur le a. En effet, si le a est égal à un, il faut qu'il soit implicite dans l'écriture donc il ne doit pas être affiché dans le terminal. Dans le cas contraire, nous affichons la valeur du a. Les deux sous-cas portent sur la solution. Si la solution unique est négative alors le moins de l'expression se transforme en plus et inversement pour la solution unique positive.



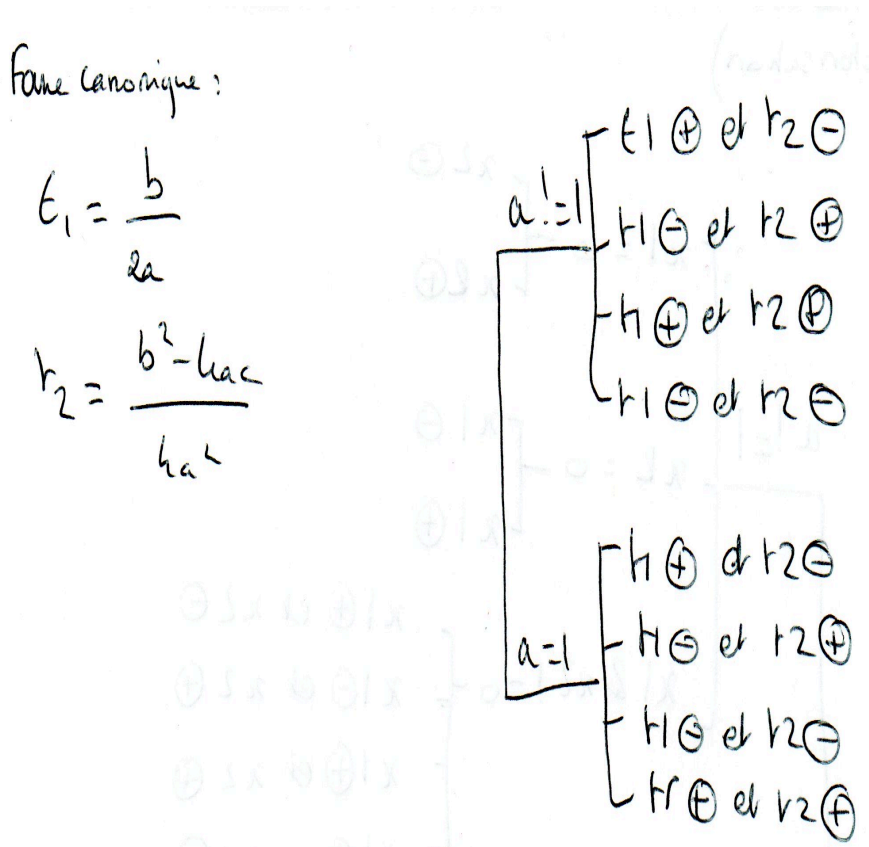
Enfin le Delta positif. Le plus fastidieux de tous, beaucoup de if...else imbriqués afin d'avoir un affichage plus réaliste. Le schéma est simple. Il y deux grands cas, trois sous grands cas, deux sous-cas. Deux grands cas comme pour le delta nul, si a est égal à un ou non. Ensuite trois sous-cas se portant notamment sur l'affichage ou non de la solution x_1 ou x_2 lorsqu'elle est nulle ou non et dans le cas contraire, on affichage les deux. Les deux sous-cas portent sur la seconde solution non nulle, si elle est négative alors c'est un plus que l'on affiche car moins suivi de moins donne plus sinon c'est un moins. Cependant pour l'affichage des deux, cela a entraîné quatre sous-cas : si l'une positive et l'autre non, inversement, si les deux positives et inversement.



Enfin nous avons aussi proposé à l'utilisateur la forme canonique du polynôme, s'écrivant :

$$a \times [(x + (\frac{b}{2a}))^2 - (\frac{\Delta}{4a^2})].$$

Là encore sur le schéma se distinguent deux grands cas, quatre sous-cas, les deux grands cas sont les mêmes que ceux du Delta nul ou positif à savoir l'affichage du a s'il est différent de un sinon non. Les quatre sous-cas sont au niveau des résultats des calculs.Nous avons fait deux petites fonctions, une qui calcule $\frac{b}{2a}$ et l'autre $\frac{\Delta}{4a^2}$. Les résultats étant des floats pour plus de précisions. En effet, si $\frac{b}{2a}$ est négatif, alors il ne faut pas laisser le plus mais il faut mettre un moins à la place et inversement. Idem pour $\frac{\Delta}{4a^2}$. Dans ce cas aussi nous avons utilisé des floats pour une plus grande précision des valeurs. Ainsi les quatre sous-cas sont : si l'un négatif et l'autre non, inversement, si les deux positifs et inversement.



2.4 Equations du troisième degré

Afin de résoudre les équations de 3ème degré, nous avons utilisé deux méthodes différentes et complémentaires. La méthode de Cardan et la méthode de Tschirnhaus.

Voici étapes par étapes comment ces méthodes fonctionnent.La fonction de résolution d'équations du troisième degré prend huit paramètres qui sont, les coefficients des membres de gauche et de droite. Les membres de droites sont ajoutés en fonction de leurs degrés à ceux de gauche afin que l'équation soit égale à 0. La deuxième étape du processus est de trouver la première des trois différentes racines du polynôme. Pour cela nous allons utiliser les deux méthodes nommées précédemment. Il faut d'abord déprécier le polynôme afin qu'il soit sous forme x^3 + cx + d = 0». Voici comment nous procédons grâce à la méthode de Tschirnhaus. :

Commençons par poser $x = t - \frac{b}{3a}$ et résolvons $f(x) = 0$:

$$a(t-\frac{b}{3a})^3 + b(t-\frac{b}{3a})^2 + c(t-\frac{b}{3a}) + d = 0$$

$$\Rightarrow \quad (t-\frac{b}{3a})^3 + \frac{b}{a}(t-\frac{b}{3a})^2 + \frac{c}{a}(t-\frac{b}{3a}) + \frac{d}{a} = 0$$

$$\Rightarrow \quad (t^3 - \frac{3b}{3a}t^2 + \frac{3b^2}{9a^2}t - \frac{b^3}{27a^3}) + \frac{b}{a}(t^2 - \frac{2b}{3a}t + \frac{b^2}{9a^2}) + \frac{c}{a}(t-\frac{b}{3a}) + \frac{d}{a} = 0$$

$$\Rightarrow \quad t^3 - \frac{b}{a}t^2 + \frac{b^2}{3a^2}t - \frac{b^3}{27a^3} + \frac{b}{a}t^2 - \frac{2b^2}{3a^2}t + \frac{b^3}{9a^3} + \frac{c}{a}t - \frac{bc}{3a^2} + \frac{d}{a} = 0$$

$$\Rightarrow \quad t^3 + (\frac{b^2}{3a^2} - \frac{2b^2}{3a^2} + \frac{c}{a})t + (\frac{b^3}{9a^3} - \frac{b^3}{27a^3} - \frac{bc}{3a^2}) = 0$$

Nous obtenons donc bien l'équation sous la forme voulue :

t^3 + pt + q = 0, avec

$$p = \frac{3ac - b^2}{3a^2}$$
$$q = \frac{2b^3 - 9abc + 27a^2d}{27a^3}$$

Nous pouvons ensuite calculer la valeur du premier Delta et de t qui seront :

$$\Delta_1 = q^2 + \frac{4p^3}{27}$$
$$t = \sqrt[3]{\frac{-q - \sqrt{\Delta_1}}{2}} + \sqrt[3]{\frac{-q + \sqrt{\Delta_1}}{2}}$$

Comme nous avons posé le changement de variable précédemment, nous pouvons aisément trouver la valeur de x1, la première des racines, avec t.

Une fois la première des racines trouvée, il est assez simple de trouver les suivantes en exprimant l'équation de la sorte :

$$f(x) = (x - x_1)(a'x^2 + b'x + c')$$

*avec

$$\begin{cases} a' = a \\ b' = b + ax_1 \\ c' = c + (b + ax_1)x_1 \end{cases}$$

Pour finir, il nous reste à résoudre l'équation du second degré tel que :

$$ax^2 + (b + ax_1)x + (c + (b + ax_1)x_1) = 0$$

2.5 Matrices

Pour finir notre projet en vue de cette première soutenance, nous avons décidé d'implémenter des calculs sur les matrices. Nous proposons à l'utilisateur de voir des exemples d'opérations de matrice, avec des rappels de cours. Nous nous sommes pour l'instant penchés sur les opérations de base comme l'addition, la soustraction, la transposé et la multiplication. L'utilisateur n'a qu'à formuler sa demande sur le terminal, comme ceci :

./matrix addition

Pour l'addition de matrice la fonction tourne avec deux boucles,l'une qui boucle sur les lignes, l'autre sur les colonnes. Nous avons des matrices à seulement une dimension ce qui fait que si nous voulons avoir l'élément à la i-ème ligne et la j-ème colonne nous faisons : $j + i * cols$, cols étant le nombre total de colonnes. A l'intérieur de ces deux boucles nous sommons les éléments et les stockons dans la matrice résultat passé en paramètre avec les deux matrices à additionner, leur nombre de colonnes et lignes.

Un exemple d'application est comme ceci :

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

Pour ce qui est de la soustraction, c'est la même chose que l'addition sauf que nous ne sommons pas mais retranchons.

Pour la transposée, c'est le jeu d'indice qui diffère. Là encore deux boucles, l'une sur les lignes, l'autre sur les colonnes. Ici nous devons mettre un élément à un emplacement différent de celui de base nous avons alors besoin de cette position : $i + j * lines$, qui permet de mettre les éléments en ligne et non en colonnes.

$$SiA = \begin{pmatrix} 1 & 4 & 5 \\ 2 & 12 & 3 \end{pmatrix} alors\ A^{\top} = \begin{pmatrix} 1 & 2 \\ 4 & 12 \\ 5 & 3 \end{pmatrix}$$

La multiplication est légèrement plus délicate. Elle nécessite trois boucles, l'une sur les lignes de la première matrice, l'autre sur les colonnes ou les lignes de l'une des deux matrices. En effet comme pour multiplier une matrice il faut que le nombre de colonnes de l'une coïncide avec le nombre de lignes de l'autre. Puis la troisième qui boucle sur le nombre de colonnes de la seconde matrice. Par ailleurs entre la seconde boucle et la troisième on initialise une variable Sum à 0 qui du coup à chaque boucle de la seconde se réinitialise. Cette variable permet de faire la somme de tous les éléments aux emplacements j-ème colonne de la première matrice et k-ème lignes de la seconde matrice. La variable ainsi remplie est alors stockée dans la matrice résultat en paramètre comme avec l'addition. La formule utilisé est la suivante :

Si $A = (a_{ij})$ est une matrice de type (m,n) et $B = (b_{ij})$ est une matrice de type (n,p), alors leur produit, noté $AB = (c_{ij})$ est une matrice de type (m,p) donnée par :

$$\forall i, j : c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Exemple :

$$\begin{pmatrix} 1 & 2 \\ 3 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 4 \\ 7 & 1 \end{pmatrix} = \begin{pmatrix} (1 \times 0 + 2 \times 7) & (1 \times 4 + 2 \times 1) \\ (3 \times 0 + 0 \times 7) & (3 \times 0 + 0 \times 7) \end{pmatrix} = \begin{pmatrix} 14 & 6 \\ 0 & 0 \end{pmatrix}$$

3 Difficultés rencontrées

En développant ce projet, nous avons été confronter à plusieurs difficultés. La première concerne la récupération des coefficients. En effet, différencier les nombres qui correspondent aux degrés et ceux correspondants aux coefficients n'est pas évident. Nous avons dû jouer sur les indices et cela n'est pas encore très optimisé, c'est à améliorer pour la prochaine soutenance mais nous l'avions prévu dans le cahier des charges. De plus, il y a plusieurs façons de noter un degré sur un terminal, nous pouvons le mettre directement en exposant ou bien à l'aide d'un ^. Pour le moment, seul le cas d'un^ est pris en compte mais nous optimiserons pour la prochaine.

Ensuite, pour ce qui est de la résolution des équations, les difficultés majeures étaient au niveau de l'affichage car en effet, dans le cas où deux signes "-" se suivent, nous avons dû les remplacer par des signes "+".

Conclusion

Nous sommes plutôt satisfait d'arriver à cette soutenance avec ce travail fourni car nous pensions en ayant l'idée de ce projet que la résolution des équations nous prendraient la majeure partie de notre temps. Cependant, nous pouvons estimer que nous avons déjà fait 70% du travail que nous pensions faire. Nous avons déjà commencé à récupérer les entrées du terminal, nous sommes capables de trouver les racines des polynômes de degré inférieur ou égal à 3. Nous sommes aussi capable d'appliquer des opérations de bases à nos matrices.

Néanmoins il reste encore beaucoup de travail. Ce que nous avons fat pour le moment n'était pas à finir pour cette soutenance et pour la prochaine, un bon nombre d'éléments viennent en plus comme notamment la réalisation d'un site web complémentaire au projet.Nous devrons en plus de cela avoir avancé toutes les autres tâches de notre projet, cela ne voudra pas dire qu'elles seront toute opérationnels mais cela voudra simplement dire que nous les auront commencé et que nous auront réfléchis sur comment les faire marcher par la suite. Le but étant bien évidemment que toute soit finie lors de la soutenance finale prévue fin mai. En attendant nous vous donnons rendez-vous à la seconde soutenance pour voir l'avancé que nous auront fait d'ici-là.