

Structure de données

Projet SSD (P)L2

Calcul Scientifique sur les polynômes

Martin Prieur de la Comble / Abel Derderian

PL2

Plan du rapport

Fonctionnement de l’algorithme	P3
1) Introduction générale.	P3
2) Saisie du polynôme.	P3
3) Traitement du polynôme.	P4
 Explication des différents modules utilisés	 P6
1) Saisie et vérification du polynôme	P6
2) Traitement du polynôme	P7
a. Découpage des monômes	P7
b. Traitements des monômes	P8
 Synthèse et conclusion	 P10

Fonctionnement de l'algorithme

1) Introduction générale

Le sujet de cette année concerne le calcul scientifique de polynôme, de manière numérique, orienté vers les mathématiques du domaine des complexes.

L'utilisateur est donc censé pouvoir saisir les composantes d'un polynôme complexe –où le polynôme directement– pour pouvoir le manipuler avec différentes opérations : addition, multiplication, division, etc...

Nous allons expliciter le fonctionnement de notre algorithme afin de détailler la manière dont nous traitons les données numériquement pour y arriver.

Le projet étant en technologie libre, nous avons décidé après concertations de le réaliser en PHP pour plusieurs raisons : une approche plus simple des traitements des données via des tableaux (simplifié par rapport au C), côté graphique extrêmement rapide à mettre en place, plus simple d'accès (une adresse web pour consulter et travailler plutôt que de devoir synchroniser nos codes sources, etc...). Ce choix a été approuvé par l'équipe encadrante avant que nous démarions le projet.

Projet accessible sur : <http://www.polynome.ovh/>

2) Saisie du polynôme

Dès l'arrivée sur la page principale, l'utilisateur est invité à saisir un polynôme complet, de type simple ou complexe (via un menu déroulant pour le type).

Exemple de polynôme simple : $-1x^{75} + 452y^{-42} + z - w^4$

Exemple de polynôme complexe : $(2x + 9iy)^{78} - (45x - 4iy)^4$

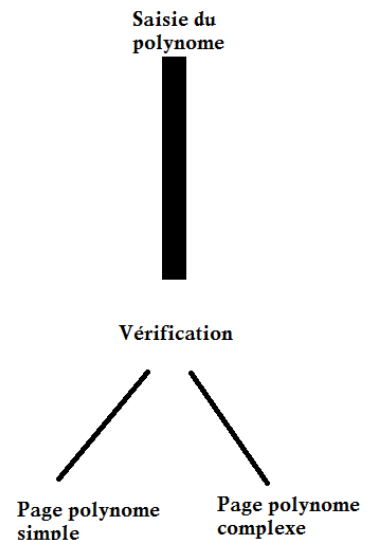
Lors de l'envoi de ce formulaire, l'algorithme va envoyer* deux variables à la même page web (index.php), qui **réagira différemment** que lors d'une visite lambda puisqu'elle détectera la présence de **variables additionnelles** à analyser.

*: Tous les détails techniques seront explicités dans une seconde partie.

Ainsi, l'algorithme va alors inclure un script de vérification dans le code afin de traiter les 2 nouvelles variables reçues, à savoir le **polynôme** (contenu dans une chaîne de caractères) et le **type du polynôme : simple ou complexe**. La vérification va effectuer plusieurs tests sur la combinaison des deux variables :

1. L'algorithme a-t-il reçu les deux variables additionnelles nécessaires ?
2. Si oui : Sont-elles conformes ? (la première doit être au moins différente de nulle et la deuxième strictement égale à « simple » ou « complexe »).
3. Si oui : Les variables sont alors transmises à deux pages différentes en fonction de leur type.

*Nous avons choisis de réaliser en plus la fonction de traitement des polynômes simples, bien que ça ne soit pas demandé dans le sujet, pour avoir une fonctionnalité en plus et surtout car son traitement ne diffère que très légèrement de celui réservé aux polynômes complexes. La transformation du script **polynôme complexe** $\leftrightarrow$ **polynôme simple** a été très rapide (moins d'une heure)*



3) Traitement du polynôme

Nous allons expliquer le traitement dans le cas d'un polynôme simple pour faire des liens plus faciles avec les explications synthétiques : le traitement pour les composantes du polynôme complexe dans notre projet est **exactement similaire**, à la seule différence que nous allons répéter certains tests.

Pour pouvoir effectuer des opérations mathématiques en programmation, nous avons besoin de connaître les composantes du polynôme reçu. Nous avons donc écrit une fonction séparée appelée dans le programme principal et qui va décortiquer le polynôme pour retourner un double tableau de 7 cases contenant les valeurs nécessaires de chaque monôme pour manipuler le polynôme. Les composantes recherchées sont :

- Le signe du monôme.
- La présence ou non d'une inconnue.
 - Si oui, la lettre de l'inconnue.
- La valeur du scalaire.
- La présence ou non d'une puissance
 - Si oui, la valeur de la puissance.
 - Si oui, le signe de la puissance.

Exemple d'un monôme : $-4x^{+47}$

Pour le polynôme complexe, la fonction de décortication des composantes est similaire, néanmoins une étape supplémentaire est faite pour déterminer deux fois les scalaires du monôme pour la partie réelle et imaginaire et l'inconnue.

L'étape suivante consiste à afficher les informations de manière ordonnée sur une page web pour que l'utilisateur dispose d'une version graphique plutôt agréable à consulter, pour qu'il puisse choisir les opérations souhaité : Addition, multiplication, division, conjugaison du polynôme, extraction des parties, évaluation en un point.

Par manque de connaissance mathématique et/ou de temps, nous n'avons pas pu finir le projet informatique, même en nous concertant avec d'autres binômes de notre promotion.

Nous avons réalisé toute la partie de saisie, traitement et analyse du polynôme pour que l'utilisateur n'ait pas à écrire individuellement les monômes ou ses composantes, et nous arrivons à la page de choix des opérations (non-disponibles, par conséquent) avec cependant toutes les valeurs nécessaires à ses opérations.

Nous avons stipulé honnêtement l'état de notre projet dans le README.txt du package.

Explication des différents modules utilisés

1) Saisie et vérification du polynôme.

Saisie du polynôme

Lorsque l'utilisateur arrive sur la page principale, il est invité à saisir son polynôme dans un formulaire, ainsi qu'à sélectionner le type de son polynôme. En validant sa saisie, il va envoyer une requête **\$_POST** (propre au PHP) contenant une chaîne de caractère pour le polynôme et une variable pouvant être égale à « **Polynôme simple** » ou « **Polynôme complexe** ».

Au début de la page d'arrivée (index.php), un script vérifie si une requête additionnelle **\$_POST** vient avec l'utilisateur. Cette condition n'est pas vérifiée la première fois, en revanche après la saisie de l'utilisateur, la requête est bien présente (ce qui vérifie la condition « Si requête est présente » ou « **if (!empty(\$_POST))** » en PHP), alors on appelle la fonction de vérification.

Cette fonction effectue quelques tests sur les deux variables, pour s'assurer qu'elles correspondent bien aux valeurs attendues, par exemple, elle vérifiera pour la première variable **qu'elle est présente et différente** d'une valeur vide ou nulle, puis elle vérifiera que la seconde variable est **strictement égale aux choix prédéfinis** : « Polynôme simple » ou « Polynôme complexe ». Suite à ça, et en fonction du type de la seconde variable, l'utilisateur aura l'affichage de son polynôme ainsi qu'un texte **lui demandant de vérifier si sa saisie est correcte**.

A partir du moment où l'utilisateur confirme sa saisie, l'algorithme va déterminer les actions à effectuer.

- Si l'utilisateur a cliqué sur « **Non** », il est simplement redirigé sur la page d'accueil.
- Si l'utilisateur a cliqué sur « **Oui** » :
 - Il sera redirigé sur un fichier dans une arborescence plus profonde : « **/simple/index.php** » pour un polynôme simple.
 - Ou il sera redirigé sur un fichier dans une arborescence plus profonde « **/complexe/index.php** » pour un polynôme complexe.

2) Traitement du polynôme

Suite à la saisie et à l'envoiJ'ax des valeurs aux pages suivantes, deux scripts de 'découpe' vont être effectués : le premier servira à découper le polynôme en la somme de ses monômes et le deuxième va analyser chaque monôme pour en sortir les composantes énoncées plus haut.

Décomposition en monômes

Pour décomposer le polynôme en une somme de monôme, nous allons nous servir des signes entre chaque partie : - ou +. Pour cela, on utilise une fonction nommée *explode*, dont l'utilité est de séparer (d'exploser) une chaîne de caractères en tableau de plusieurs colonnes. Elle sépare la chaîne aux endroits où se trouve le caractère fourni.

Exemple : $\$tab = explode('a', "Je trouve ça très hilarant")$;

Le tableau $\$tab$ vaudra :

$\$tab[0] \rightarrow$ 'Je trouve ç'

$\$tab[1] \rightarrow$ ' très hil'

$\$tab[2] \rightarrow$ 'r'

$\$tab[3] \rightarrow$ 'nt'

On se sert donc dans un premier temps de cette fonction pour décomposer le polynôme avec le signe - puis avec le signe +. Cependant, voilà : que se passe-t-il si une puissance est négative (sait-on jamais) où qu'à l'intérieur d'un monôme complexe une des parties l'est ?

Nous allons utiliser la fonction **substr(\$variable, [position])**. Cette fonction donnera la valeur du caractère à la position [position] dans la chaîne. Si on décide de mettre 3 à la place de position, la valeur retournée est le 3ieme caractère de la chaîne. Par ailleurs, si nous mettons une position négative, le compte remontera en partant de la fin du monôme précédent (= le tableau inférieur).

Ainsi, nous pouvons vérifier le caractère suivant ou précédant la césure de la fonction **explode**. Si le caractère précédent est différent de «) » et le suivant de « (», alors ce n'est pas une bonne séparation, et on regroupe les deux monômes (le précédent et le suivant) en un seul.

Exemple : $(3-4iy)^{(2)}(3iy)^{(4)}$ → Il n'y a que le signe en vert qui correspond bien aux conditions, donc c'est le seul endroit qui divisera le polynôme en deux monôme.

Ainsi, la fonction décrite ci-dessus, nommée **decouper_polynome()**, va retourner un tableau contenant n colonnes, correspondant au nombre de monômes dans le polynôme.

Suite à ça, nous allons décortiquer les différents monômes pour obtenir leurs composantes.

Décomposition des monômes en composantes.

Nous avons besoin de connaître les composantes de chaque monôme pour pouvoir manipuler le polynôme intégralement, et à cette étape nous disposons d'un tableau contenant l'ensemble des monômes. Ainsi, nous allons effectuer une boucle « Tant que » pour analyser chaque monôme avec une fonction codée par nos soins. Cela se présente comme cela :

Tant que (monome n°i) n'est pas nul

tableau2 = determiner_monome(monome);

i++;

Fin tant que.

Ainsi, chaque monôme sera envoyé à la fonction, et le résultat sera un autre tableau. On stockera donc le résultat dans un tableau à double indice : `$tableau[i][j]`.

- i sera le numéro du monôme dans le polynôme (de la gauche vers la droite).
- j sera la colonne spécifique à une des composantes (signe, scalaire, etc...).

La fonction `determiner_monome` va déterminer les composantes énumérées plus haut :

Puissance

Tout d'abord, la puissance. Il va déterminer s'il y a une puissance ou non. Avec la fonction `substr_count($variable, "^")` (qui renvoi une valeur booléenne), on détermine s'il y a une puissance. Si non, 3 variables ne seront pas à définir :

1. La présence de la puissance
2. Le signe
3. Le scalaire

Si la puissance est bien là, on la sépare du reste avec la fonction **explode** précédemment vue (et on enlève la parenthèse fermante avec une fonction nommée **str_replace**, qui remplace une occurrence dans une chaîne par une autre, dans notre cas on remplace ')' par ").

Le signe de la puissance : On cherche dans la variable qui contient la puissance une occurrence égale à un – ou un +. Si il n'y en a pas, on choisira le + par défaut.

Le scalaire de la puissance : Dans la variable contenant la puissance, en enlevant le signe et la parenthèse fermante, il ne reste par définition que le scalaire de la puissance.

L'inconnue du monôme : Via la même fonction (faite par PHP) `substr_count` et une regex cherchant les lettres de a à z et de A à Z, on obtient la variable. Si on n'en trouve pas, on définit l'inconnue à 0.

Le signe du monôme : De la même manière que pour l'inconnue, le procédé est le même, on cherche cependant un – ou + cette fois-ci.

A la fin de la fonction, on retourne un tableau de valeurs à 1 dimension qui ira se greffer dans le tableau à 2 dimensions principal, lequel se remplit au fur et à mesure que la boucle des monômes s'exécute. On obtient donc un tableau à 2 dimensions contenant les différentes valeurs pour chaque monôme (respectivement colonnes et lignes).

Synthèse et conclusion

Notre projet n'étant malheureusement pas fini, nous n'avons pas pu effectuer les opérations arithmétiques sur C des polynômes complexes obtenus (même si l'interface graphique pourrait laisser penser le contraire). Ce projet, bien qu'imparfait, nous a appris à tourner longuement un problème sous plusieurs angles pour obtenir un moyen de « numériser » nos actions tel qu'extraire une valeur particulière.

Nous pensons peut-être à continuer le projet plus sérieusement pendant les vacances ou l'année prochaine, et passer le programme en open-source pour obtenir des participations. L'idée serait de pouvoir obtenir un site utilisable par n'importe quel collégien / lycéen / étudiant.