APIthagore
Cahier des charges

Epiquation

Fernando Borja
Lucas Rangeard

Thomas Boucard

14 mars 2017

Table des matières

1 Introduction

Le projet Epiquation est un solveur d'équation, qui a pour but de faciliter le travail et les calculs avancés et simples. Le groupe APIthagire formée pas trois étudiants de API (année préparatoire pour le cycle ingénieur) s'est assez facilement formé du fait d'avoir déjà travailler plus d'un semestre ensembles. Notre projet étant assez ambitieux pour des étudiants de SPE, la phase de recherche et création d'algorithme a été une étape importante et nécessaire avant toute implémentation. On a donc penser à plusieurs techniques et après quelques recherches nous avons opter à utiliser les arbres binaires comme basse de notre solveur. Ce rapport détaille le travail effectué depuis la naissance du projet. Nous verrons que ce dernier a bien avancé et respecte le rendu de la première soutenance, et a même dépassé certains de nos objectifs initiaux pour cette soutenance. Dans un premier temps nous présentons le projet de façon général puis une présentation plus détailler partie par partie.

2 Présentation du projet

Alors que nous progressons dans le programme en mathématiques et que notre niveau évolue, la nécessité de pouvoir rapidement vérifier la justesse de nos calculs s'est faite sentir. Il nous est alors venu à l'esprit l'idée de développer un solveur mathématique qui puisse nous aider lors de notre formation, et nous donner plus d'aisance avec la matière.

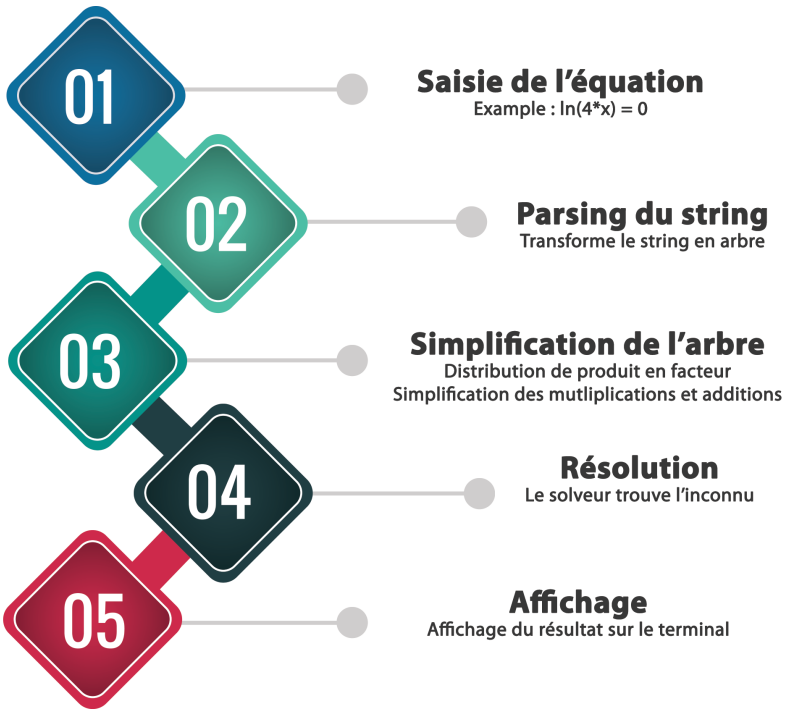

FIGURE 1 – Étapes de traitement

Sur la figure 1 les étapes de traitement du logiciel sont explicitement expliquées. Tout d'abord la saisie de l'équation sur le terminal suivi par le parsing qui s'occupera de créer des arbres avec les priorités de calculs, puis la simplification de l'arbre qui va simplifier le plus possible l'équation pour passer à l'étape quatre qu'est la résolution de l'équation, puis en dernière étape on gère l'affichage on montrent explicitement le résultat de la variable.

2.1 Objectifs soutenance 1

Pour la soutenance du 14/03/2017 nous avons planifiées plusieurs tâches et objectifs, voici la liste de nos objectifs étudiés et développés pour cette soutenance :

1. Parsing
2. Solveur
3. Fonctions affines
4. Puissance et racine
5. Trigonométrie
6. Site Web

Aujourd'hui tous les points cités plus haut sont fonctionnels et affichent des bons résultats, les algorithmes implémentés ne sont pas encore optimisés.

3 Répartition des tâches

Soutenance (13/03/2017)			Fernando	Lucas	Thomas
Analyse					
	Parsing		x		x [1]
	Structure		x		x [1]
	Solveur				
		affine			x [1]
		puissance et racine		x [1]	
		trigonométrie		x [1]	x
Développement					
	Parsing		x		x [1]
	Solveur				
		affine			x [1]
		puissance et racine		x [1]	
		trigonométrie		x [1]	x
	Site Web		x [1]		x

1. chef de la tâche

4 Structure de données

4.1 Les unions

4.1.1 Le type

L'utilisation d'une variable de type (void*) pour stocker les données de l'arbre (sa structure sera expliquée plus tard) nous a poussé à utiliser une énumération pour définir les différentes données que celui-ci vas être amené à stocker. Cet ensemble contient les valeurs suivantes :

— OPERATOR
— FUNCTION
— VARIABLE
— VALUE

4.1.2 Les opérateurs

Les opérateurs ayant un nombre de valeurs défini, nous avons aussi choisi de les représenter sous la forme d'une énumération contenant les valeurs ci-après :

— PLUS
— MINUS
— TIME
— UNKNOW

On peut remarquer l'absence de l'opérateur de division qui bénéficie d'un traitement spécifique expliqué plus tard dans ce rapport. Pour les spécificités de certains de nos algorithmes nous avons aussi rajouté la valeur "UNKNOW".

4.1.3 Les fonctions

Les fonctions sont gérées de la même manière que les opérateurs. Ainsi une liste de fonctions à été défini de même qu'une inconnu appelé "UNKNOW_F". Celle-ci contient :

- LN
- EXP
- POW
- SQRT
- COS
- ACOS
- SIN
- ASIN
- TAN
- ATAN
- UNKNOW_F

4.2 Les structures

4.2.1 L'arbre

L'arbre est la parti fondamentale de notre projet. C'est grâce à lui que nous pouvons prioriser les opérations. Sa structure est la suivante :

```
1  enum e_type
2  {
3    OPERATOR,
4    FUNCTION,
5    VARIABLE,
6    VALUE
7  };
8
9  struct          s_tree
10 {
11     struct s_tree *left;
12     struct s_tree *right;
13     enum e_type   type;
14     void          *data;
15 };
```

FIGURE 2 – Struct Tree

Elle contient un champ type qui permet de connaître le type du champ data. Ce dernier peut contenir des informations à propos d'un opérateur, d'une fonction, d'une variable ou un nombre.

4.2.2 Les fonctions

Pour pouvoir gérer les fonctions nous avons aussi utilisé une structure. Celle-ci vas nous permettre de stocker divers information telle que le paramètre utilisé pour les fonctions puissance et racine. Ainsi que les champs pow et mult qui contiennent respectivement la puissance et le coefficient multiplicateur de la fonction. Sa structure est défini ci-après.

```
1  struct s_function
2  {
3    enum e_function *function;
4    float           param;
5    int             power;
6    float           mult;
7  };
```

FIGURE 3 – Struct fonction

4.2.3 Les variables

Les variables respectent le même principe que les fonctions à l'exception du paramètre qui n'est pas présent.

```
1  struct          s_variable
2  {
3    char          name;
4    int           power;
5    float         mult;
6  };
```

FIGURE 4 – Struct variables

4.2.4 Les listes

Les listes utilisées sont de simple listes chaînées utilisant une sentinelle.

```
1  struct s_list {
2    struct s_list *next;
3    struct s_tree *tree;
4  };
```

FIGURE 5 – Struct list

5 Les algorithmes

5.0.1 Le parsing

Le parsing consiste à rendre utilisable une chaîne de caractère pour notre programme. Ce parsing transforme donc une chaîne de caractère saisie par l'utilisateur en un arbre ayant pour racine le signe égal. Le fils droit(resp. gauche) sera la partie droite(resp. gauche) de l'équation. Ces arbres seront créés de manière à respecter les priorités d'opérations. L'ordre de priorité de traitement étant :

1. La recherche d'opérateurs prioritaire(si absence de parenthèses)
2. Recherche de fonctions
3. Gestion des parenthèses
4. Nombres ou variables

Un traitement est effectué pour les divisions afin de les transformés en multiplications. La fonction clean_string sera nécessaire afin de rendre l'équation saisie compréhensible pour le parseur. Pour l'instant, il est nécessaire d'inclure une variable et un signe égale dans la chaîne saisie.

5.0.2 La simplification des additions et des multiplications

La simplification des multiplications (resp. additions) passe tout d'abord par un parcours d'arbre qui lors d'une rencontre avec un signe multiplier (resp. additionner), commence a construire une liste avec n'importe quel type sauf une valeur. Si une valeur est rencontrée, elle est multipliée (resp. additionnée) au coefficient afin de le garder lors de la remontée récursive. l'arbre est ensuite reconstruit.

5.0.3 La reconstruction de l'arbre

La reconstruction de l'arbre récupère la liste résultant de la simplification. Pour reconstruire une suite de multiplications, il crée des nœuds contenant l'opérateur multiplier possèdent en feuille les valeurs de la liste. La première feuille étant multiplier par le coefficient. La reconstruction de l'addition fonctionne de manière similaire à la multiplication à ceci prêt que le coefficient est ajouté en feuille de l'arbre reconstruit.

5.0.4 Multiplication d'un arbre

Cette étape correspond entre autre a un développement. Il y a 5 cas possibles :

- La rencontre avec un signe "+", l'appel recusif se fait sur les deux fils.
- Lors de la rencontre d'un signe "*", le rappel recursif ne se fait que sur le fils gauche.
- La rencontre avec une valeur multiplie celle-ci avec le coefficent retenu lors du parcours recursif.
- La rencontre avec une variable ajoute le coefficient à la structure associée à la variable.
- De même lors de la rencontre avec une fonction, le coefficient dans la structure est modifé.

5.0.5 Le solveur

Construit deux listes contenant les différents sous arbres séparés par une addition ou une multiplication, elle doivent toutes être séparées par le même symbole. Ces listes sont ensuite parcouru, les nœuds contenant des variables sont réunis dans une liste tandis que la valeur des autres nœuds sont calculées. Le résultat est ensuite calculé et renvoyé à l'utilisateur.

5.0.6 le gestionnaire d'erreurs

Le gestionnaire d'erreurs affiche simplement une chaîne de caractères suivant un nombre (une variable statique) donné.

6 Problèmes rencontrés

Lors de la phase de développement, plusieurs problèmes ont été rencontrés. La recherche d'algorithmes efficace nous a demandé énormément de temps. La résolution de bugs à elle aussi été chronophage.

Voici la liste des problèmes majeurs rencontrés.

6.1 Création de l'arbre

La création de l'arbre a été possible grâce a un champ type et un pointeur sur data ajouté à la structure de l'arbre.

6.2 Gestion des soustractions

Il nous a été nécessaire de simplifier l'arbre le plus possible ainsi nous avons eu l'idée d'appliquer la soustraction sur le fils droit de l'arbre afin de remplacer les nœuds de celui-ci par des additions.

6.3 Portée des soustractions

La soustraction nous a posée un second problème lors de la phase de testes. En effet, lorsque l'équation saisie commençait par un signe moins, le résultat affiché n'était pas celui escompté. Il a fallu effectuer des vérifications sur les algorithmes afin que l'opération s'effectue de la manière dont nous le souhaitions.

6.4 Gestion des divisions

Par un procédé similaire à celui de la soustraction, la division dans l'arbre s'opère en remplaçant cette opération par une multiplication a la puissance -1. Il a bien sûr fallu s'assurer que le nombre n'était pas nul.

6.5 Opérations sur l'arbre

Enfin, l'un des plus gros problème fut la gestion des opérations sur l'arbre. Nous avons finalement opté pour une cohabitation entre arbres et listes. Si nous avions seulement utilisé la manipulation des arbres, le nombre de cas à prévoir auraient été bien plus nombreux.

7 Technologies utilisées

Certaines contraintes et besoins quant aux technologies utilisées. On du Les contraintes sont les suivantes :

- Le projet est fait sous **Linux**.
- Il a été codé en **C**.
- Le projet fonctionne sur les **machines de l'école**.
- Un **site** est présent afin de montrer l'état d'avancement du projet.

De même nous avons utilisé les librairies standard suivantes :

- err.h.
- math.h.
- regex.h
- stringh.h.
- stdlib.h.
- stdio.h.
- sys/types.h

7.0.1 Le Regex

Nous avons utilisé pour ce projet la technologie regex qui permet de détecter des motifs dans des chaînes de caractères.

Disponible sur les ordinateurs de l'école par le biais de la bibliothèque Posix « regex.h »cette technologie nous a permis d'analyser les chaines de caractère.

8 Site Web

Le site Web est une partie importante et essentielle du projet, il sert de vecteur majeur de communication avec la communauté des développeurs mais aussi au public. Celui-ci sert à présenter le projet de façon global mais aussi de façon précise si le visiteur veut en savoir plus. Nous avons opté pour un site "One-page" qui permet d'avoir tout le contenu que nous avons besoin sur une unique page, nous avons de même opté pour un site assez épuré mais aussi attirant et facile d'utilisation.

Sur celui-ci nous présentons le projet, le groupe, notre répertoire "Github" de la version actuel du projet, le cahier des charges et tous les rapports de soutenances sont disponibles. Puis nous essayons de mettre régulièrement à jour le site avec des news et aussi en mettant à jour les animations rendant celui-ci assez interactif.

Nous avons opté à l'héberger sur "Github" car grâce à nos adresses mail de l'EPITA nous avons des comptes professionnels gratuits nous donnant assez à la partie "Pages" de Github, ainsi l'URL du site est :

https ://fadao23.github.io/webepiquation/

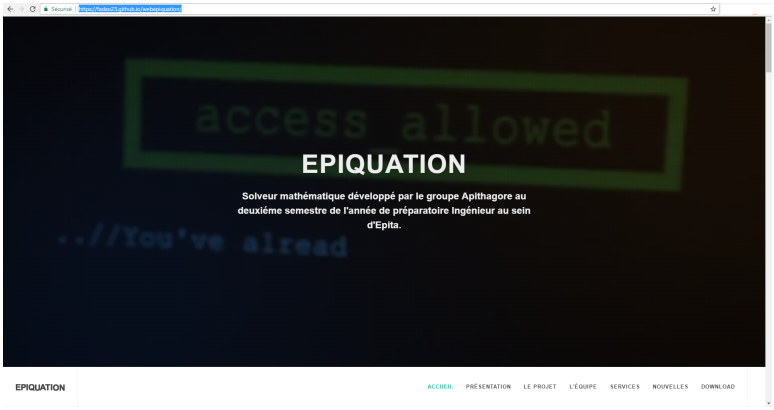

Figure 6 – Index site Web

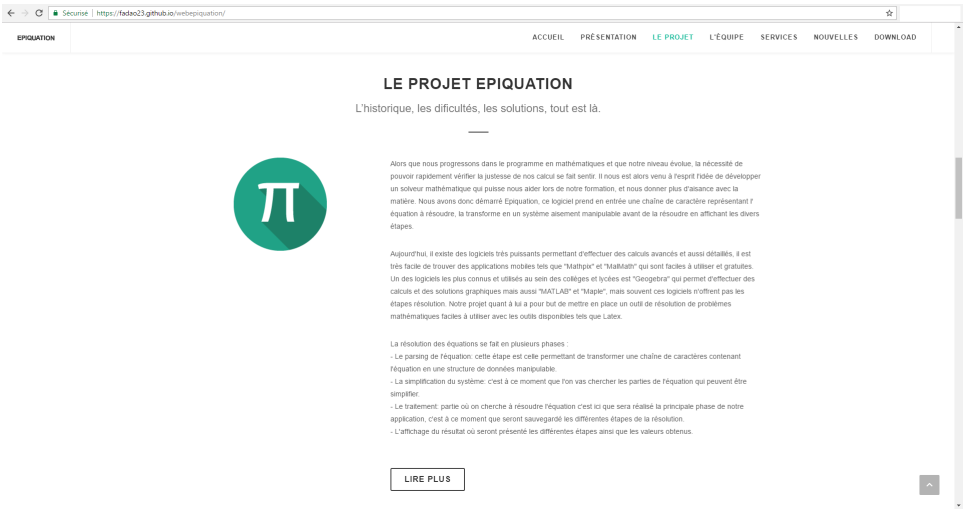

FIGURE 7 – Présentation projet

Sur la partie présentation du projet nous avons fait une simple présentation avec l'état de l'art sur les logiciels et application mobiles disponibles sur le marché actuellement. Si l'utilisateur souhaite avoir plus d'informations, nous avons inclus un bouton "Lire plus" qui redirige vers le cahier des charges du projet géré avec "PDF js" de chez Mozilla.

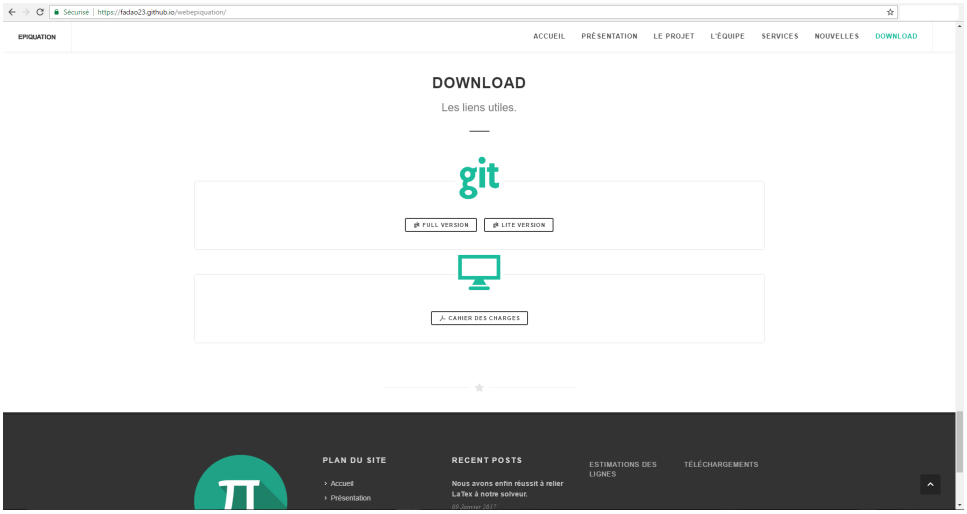

FIGURE 8 – Téléchargements

9 Objectifs soutenance N°2

Soutenance (24/04/2017)			Fernando	Lucas	Thomas
Analyse					
	Solveur				
		logarithme et exponentiel		x	x [1]
	Système		x		x [1]
	Derivée			x [1]	x
	Primitive		x [1]		x
Développement					
	Solveur				
		logarithme et exponentiel		x	x [1]
	Système			x	x [1]
	Polynome		x [1]		x
	IHM			x [1]	
	Site Web		x [1]		

Pour la prochaine soutenance qui se déroulera dans la semaine du 27/04/2017 nous avons plusieurs objectifs, tout d'abord nous allons commencer à travailler sur un aspect visuel de notre logiciel avec un début d'IHM, ensuite nous souhaitons avoir des résolutions de systèmes et polynôme, il faut savoir que les fonctions logarithmiques et exponentielles sont déjà prisses en charge par notre logiciel. Nous avons de même pour objectif d'optimiser nos algorithmes mais aussi d'avoir une plus grande gestion des erreurs avec un parsing améliorer et des simplifications d'arbres plus efficaces.

1. chef de la tâche

Table des figures