Acknowledgements

This thesis and everything would not be possible without Mr Sebastien MENA and Mr Gille LANDSMANN’s 6 month industrious support, instruction, and guidance, which I cannot be more grateful for.

I would also like to thank Professor Etienne BORDE and Professor Remy SHARROCK for their excellent teaching and advisement.

My graduate life has been blessed because of my many friends. Even when things get tough, they never stop caring and supporting me.

Toulouse, january 30th 2015

Abstract

Lorem

Keywords

Résumé

Mon parcours à Télécom ParisTech m’a amené à découvrir de nombreuses technologies nécessaires en développement du logiciel. Afin de pouvoir me familiariser avec ces technologies dans un cadre plus concret que ceux étudiés en cours, je choisis de faire un stage au sein du groupe Airbus où je dois réaliser un projet qui tire profit des connaissances acquises pendant mes deux ans d’études.

Le projet que j’ai choisi est un logiciel pour l’analyse des données anémométrique. En effet, l’objectif est de fournir un logiciel gratuit pour analyser et manipuler des fichiers qui génèrent par les tests de l’avion et par le bureau d’étude aux collaborateurs du groupe. Cela demande aussi de choisir une architecture de code permettant le développement en module de code réutilisable. Ce stage est très orienté développement en objet, qui sera l’activité prépondérante pendant toute sa durée.

Ma mission consiste tout d’abord à analyser des outils existants sous linux et les formats de fichier de type de la loi de command et de type de test pour définir l’exigence du projet, et ensuite de concevoir un IHM windows sous l’environnement de développement PyQt. Pour ce faire, j’ai dû utiliser le langage de programmation Python en utilisant la bibliothèque Qwt pour réaliser le tracement des courbes.

Les problématiques de réalisation et d’organisation propres à ce projet sont à la hauteur de ses enjeux, car il a pour but d’être réutilisé dans une structure plus grande, contenant par exemple faire le découpage sur des iso et super-iso, le dégradé de couleur sur les courbes, le suivi des points sur la table à tracer... et doit donc s’imposer comme une base solide sur laquelle d’autres utilisateurs pourront s’appuyer.

Mots-clés

Tracement, découpage, suivi, édition

Table des matières

[Acknowledgements v](#_Toc410228071)

[Abstract vi](#_Toc410228072)

[Keywords vi](#_Toc410228073)

[Résumé vii](#_Toc410228074)

[Mots-clés vii](#_Toc410228075)

[Chapter 1 Introduction 9](#_Toc410228076)

[1.1 Title 1.1 9](#_Toc410228077)

[Chapter 2 Contexte du sujet 11](#_Toc410228078)

[2.1 An 11](#_Toc410228079)

[2.1.1 Title 2.2.1 11](#_Toc410228080)

[Chapter 3 Flot de modélisation 13](#_Toc410228081)

[3.1 L’architecture de logiciel 13](#_Toc410228082)

[3.2 Spécification du cahier des charges 14](#_Toc410228083)

[3.3 Présentation des classes 15](#_Toc410228084)

[3.3.1 Gestion de l’espace de travail 16](#_Toc410228085)

[3.3.2 Gestion des fichiers importés 17](#_Toc410228086)

[3.3.3 Gestion des actions utilisateur. 21](#_Toc410228087)

[3.3.4 Vérsion finale de modélisation. 22](#_Toc410228088)

[Chapter 4 Fonctionnalités important 23](#_Toc410228089)

[4.1 Ordinate = f (abscissa) 24](#_Toc410228090)

[4.2 Ordinate = f (abscissa, iso) 26](#_Toc410228091)

[4.3 Ordinate = f (abscissa, iso, siso) 29](#_Toc410228092)

[4.4 Tracement d’un label 32](#_Toc410228093)

[4.5 Labels Superposition 34](#_Toc410228094)

[4.6 Labels and courbe csv superposition 36](#_Toc410228095)

[4.7 Manipulation des courbe 37](#_Toc410228096)

[4.8 Suivi & Affichage des points sur traceur 38](#_Toc410228097)

[4.9 Edition des courbes tracé par les labels 39](#_Toc410228098)

[4.10 Sauvgardé de la session de travail d’utilisateur 40](#_Toc410228099)

[Chapter 5 Conclusion 41](#_Toc410228100)

[5.1 Tâches réalisés 41](#_Toc410228101)

[5.2 Developpement en Future 41](#_Toc410228102)

# Introduction

Lorem ipsumdo

.

## Title 1.1

Lorem ipsumdolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulumut, placerat ac, adipiscing vitae, felis. Curabitur

# Contexte du sujet

Dans ce chapitre, on va d’abord parler d’abord c’est quoi anémométrie et clinométrie, et puis, un cas de test va être présenté afin de vous monter pourquoi on a besoin de cet outil.

## An

Nam lacus liber

.

### Title 2.2.1

Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi

# Flot de modélisation

Avant de commencer à expliquer ma conception, on va tout d’abord analyser une figure qui est une partie d’un fichier qui a été généré par un équipement spécifique sur avions, on peut trouver sur la figure, les paramètres et sont valeurs sont mémoriser cologne par cologne en type de fichier csv. En fait les capteurs vont faire acquisition des paramètres huit fois par second, est à la fin de test, on obtient un fichier qui sauvgarder les déroulements des données pendant la durée des vols.

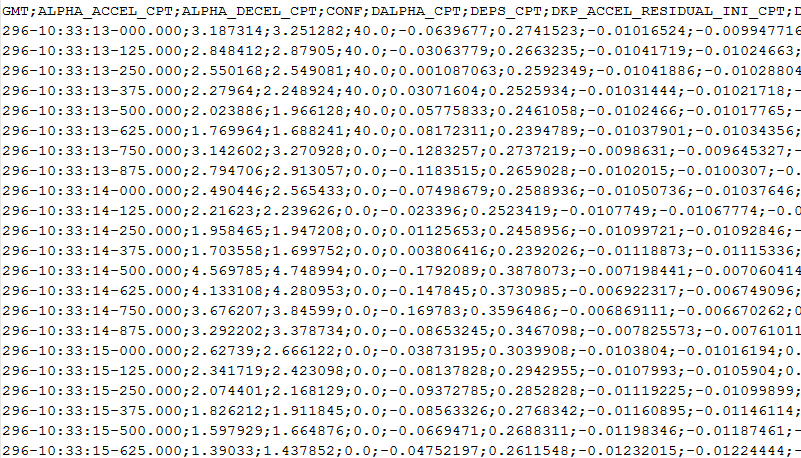


Figure 3:0 une partie de fichier csv

L’Objectif de l’outil, c’est de lire les fichiers de ce type, est analyser les relations entre les paramètres via tracer les courbes en 2D en ajoutant des operations spécifiques. Après plusieurs fois de comparer et consulter, on a choisi d’utiliser le librairy QWT pour réaliser ce traceur.

À partir de cela, je commence à modéliser l’architecture de mon logiciel.

## L’architecture de logiciel

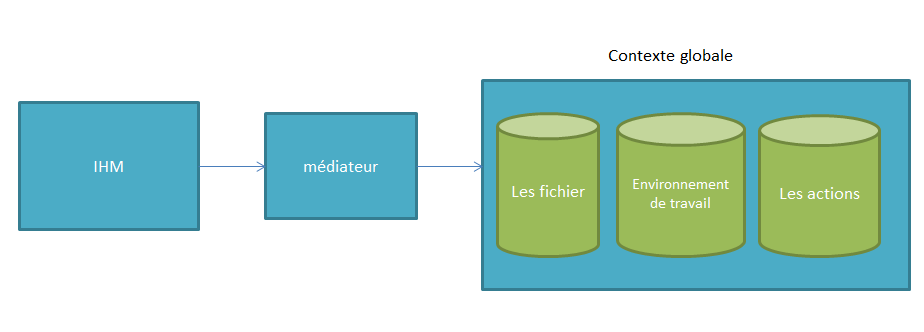
Dans ce logiciel, il faut tout d’abord avoir un ensemble d’IHM qui sert à manipuler par utilisateur, et puis, on doit crée une base qui sauvgarder les données dans les fichiers, les sessions de travail d’utilisateur, et les chaînes de actions. Enfin, j’ai crée un module qui jouer le rôler comme un médiateur qui transfère les requêts d’utilisateur pour accéder au contexte globale. 

Figure 3:1 l’architecture de logiciel

## Spécification du cahier des charges

Le diagramme de cas d’utilisation ci-dessous donne une vue fonctionnelle du analyseur, il nous montre des principales fonctionnalités du logiciel selon point de vue d’un utilisateur final.

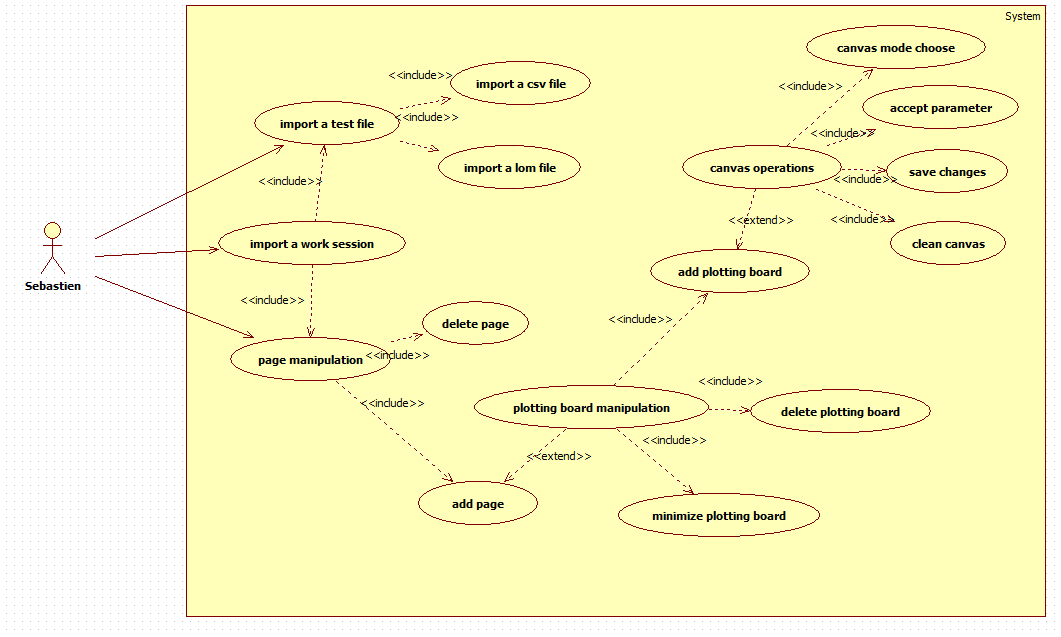


Figure 3:2:0 diagramme de cas d’utilisation

Supposons que l’on est un utilisateur de ce logiciel, on peut créer les pages sous espace de travail, puis crée les tables à tracer, qui sont l’unité graphique de base pour tracer les courbes et coupers les courbes, sous une page. Et on doit aussi avoir des options pour importer les fichiers de test ou fichiers de lom. Et enfin, une option afin de réstaurer une session de travail est indispensable pour cet outil. Dans ce document, je ne vais pas concentrer beaucoup sur le développement des IHM, car c’est très clair dans mon code ressource, dans ce document de developpement, je détaillerai étape par étape le architeture global de ma modélisation, et les fonctionnalités dans un ‘plotting board’. Car ce sont les deux parties le plus essentielles de mon logiciel.

Figure 3:2:1 l’ensemble des vues du logiciel

## Présentation des classes

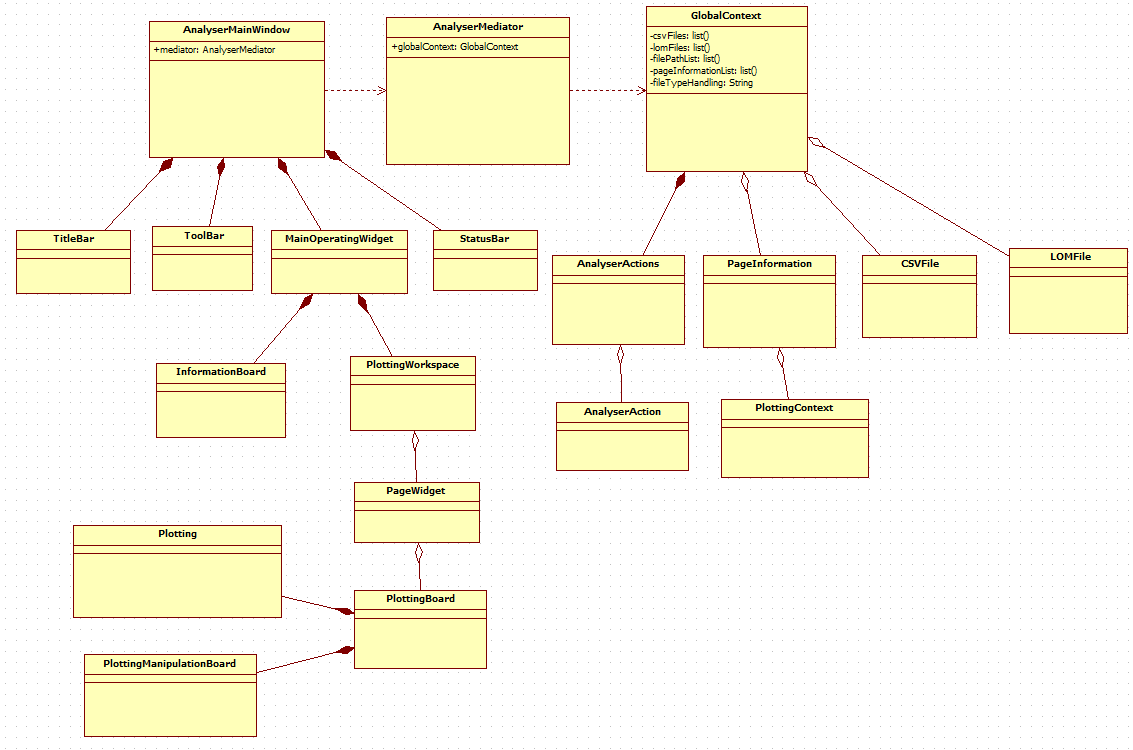


Figure 3:3:0 diagramme de classe d’analyse

Vous trouvez en ci-dessus le diagramme de classe d’analyse qui a été vite fait après avoir inspiré sur l’architecture de logiciel (figure 3 :1), On peut bien remarquer que la classe médiator découpe l’ensemble de vues et l’ensemble de modèle en deux parties. Et maintenant, je vais expliquer et remplir successsivement comment j’ai obtenu mon architecture du logiciel final en quatre étapes.

### Gestion de l’espace de travail

Dans l’exigence de cet outil, l’analyseur doit avoir une espace de travail qui peut ajouter des pages selon besoin d’utilisateur, sur chaque page, on peut créer 6 tables à tracer en maximale. On va récupérer les deux parties dans le diagramme de classe (Figure 3:3:0) rélié à cet aspect, dans la figure ci-dessous, on peut trouver une architecture arborescence des composants graphiques à gauche dans l’ensemble des vues et une architecture arborescence pareil pour les sessions de travail dans le modèle.

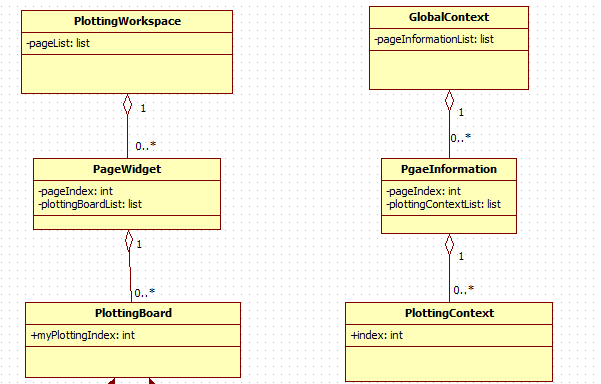
Grâce à cette conception, à chaque fois que utilisateur créer une page, le médiateur va crée, dans le modèle, un objet qui sauvgarder tous les informations concerne du page qui vient de créé. Quand utilisateur crée une table à tracer sous une page, le médiateur va créer une ‘plotting context‘, qui sauvgarde l’ensemble des données à tracer, sous le page information qui corresponds à pageWidget.

Figure 3:3:1:0 espace de travail dans le vue et le modèle

Voici un exemple. Utilisateur veut crée 4 tables à tracer en page 0, et 3 tables à tracer en page 1. La figure ci-dessous montre la procédure qui a été fait au dérrière. Donc dans le contexte globale, l’espace qui représente l’environnment de travail, on crée deux planche pour page 0 et page 1, et puis dans page 0, 4 ‘plotting context’ sont créé successivement. Pareil pour le page 1.

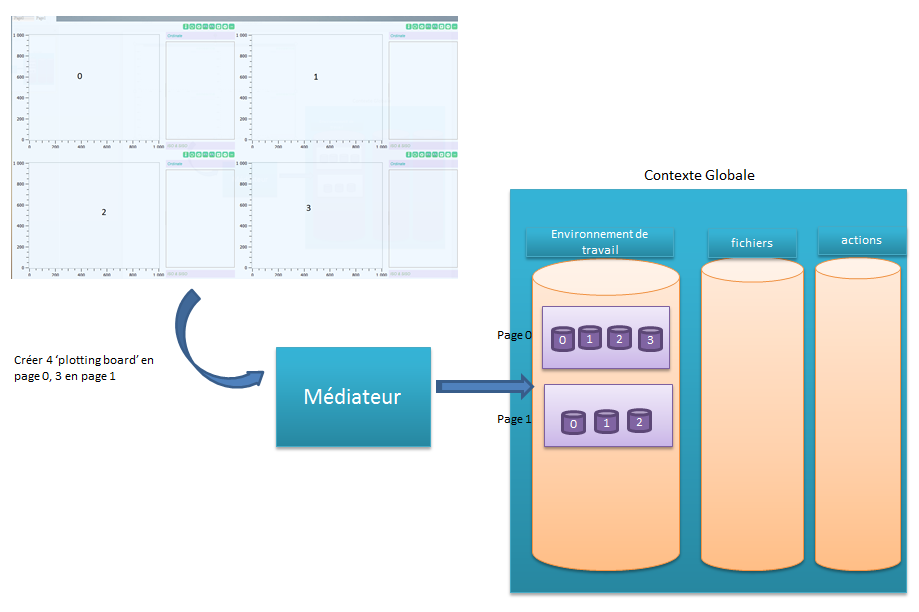


Figure 3:3:1:1 un cas test de l’espace de travail

### Gestion des fichiers importés

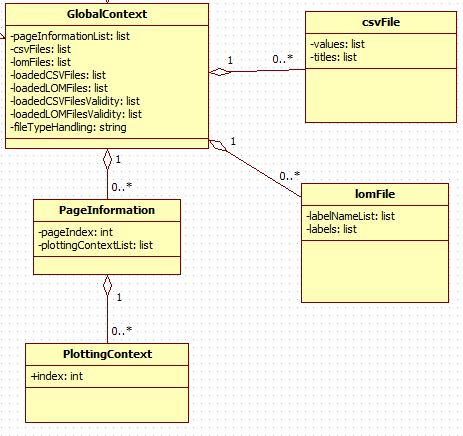
Comme nous le savons, les fichiers de test (csv) et les fichiers de la loi de command (lom) est la base de cet outil, tous les fonctionnalités que l’on va réaliser dans cet outil sont basé sur des données dans ces deux type de fichiers. Donc il faut crée une espace dans le contexte globale pour sauvgarder tous les contenus dans un fichier. On peut évoluer le contexte global (modèle)) comme la figure ci-dessous. Maintenant, dans le contexte global, on a une list des objets de fichier de type csv et une autre list des objets de type lom.

Figure 3:3:2:0 environnement de travail et des fichiers dans le modèle

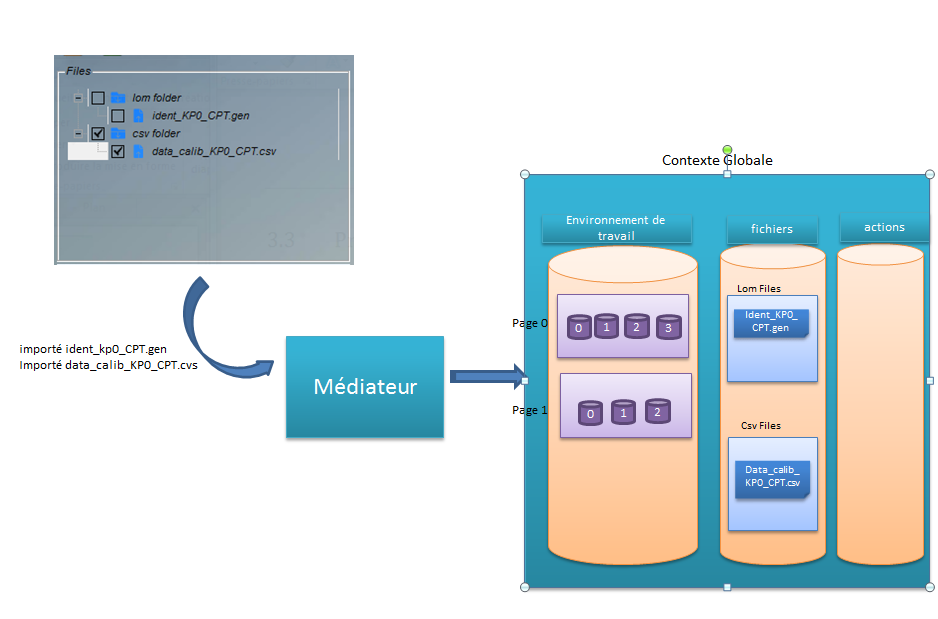
Voici un exemple : utilisateur a importé deux fichiers, alors, le médiateur a récupéré le chemin vers ces deux fichiers, et instanciers des objets de fichiers selons le type, et puis mettre les objets de fichier dans le partie qui gère les fichiers du contexte globale.

Figure 3:3:2:1 un cas test de l’environement de travail avec des fichiers

Une fois que plusieurs types de fichiers ont été bien importés dans le contexte global. Comment on va différencer les deux types dans une table à tracer?

En fait, un label dans un fichier de LOM est une loi de command qui a été calculé par le bureau d’étude. et le mission de notre équipe est de tracer des paramètres dans les fichiers test en comparant avec le loi du command afin de trouver si il y a des erreurs entre le cas pratique et le cas de théorique. Donc ce n’est pas la même chose au niveau de façons de tracer.

Ce que je vais réaliser c’est que, je prends un paramètre qui viens d’un fichier, et mettre sur la table par le souris. Et puis, si ça vient du fichier de test, donc je vais demande à chaque fois comme quel axe utilisateur veut définir. En fin, si un graph a possède déjà un absciss et un ordinate, un courbe y = f(x) va être tracé sur la table à tracer automatiquement.

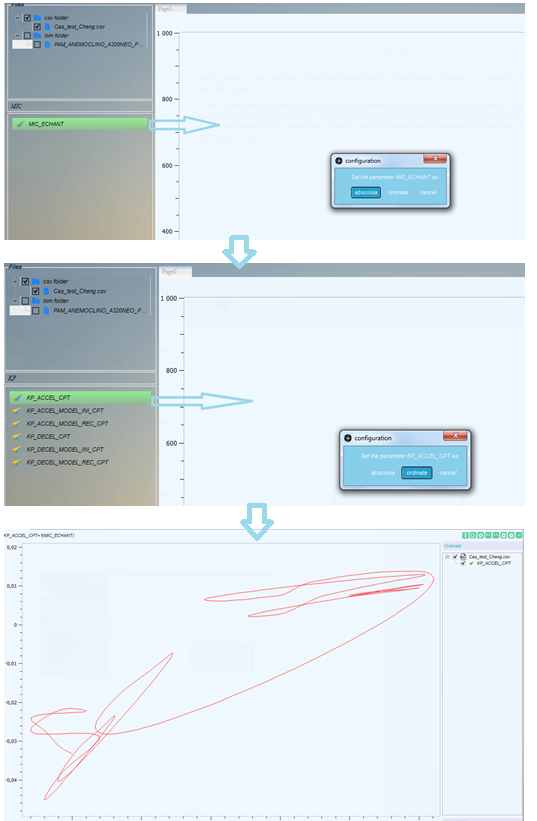


Figure 3:3:2:2 la procédure de tracer une courbe y = f (x)

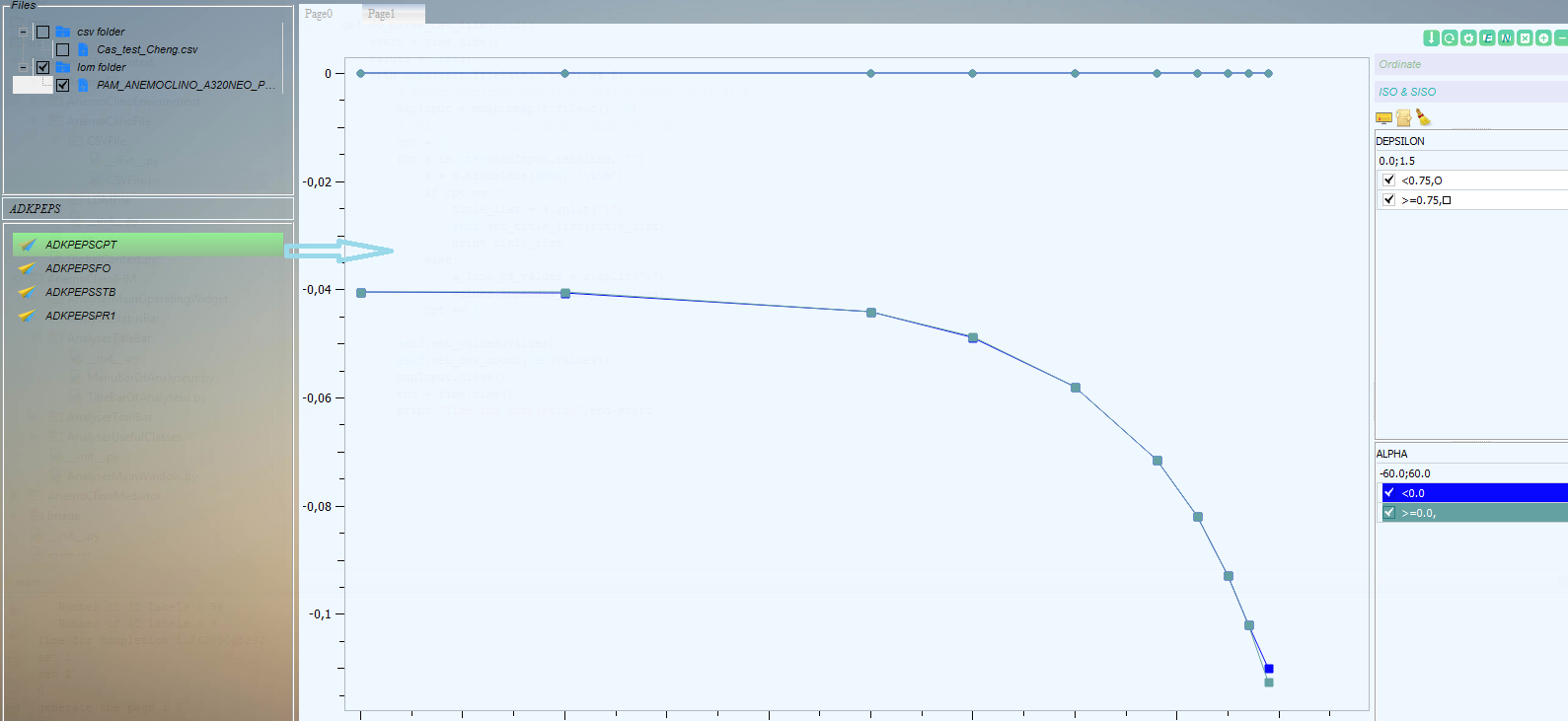
Si le paramètre est un label qui vient du fichier de LOM, une courbe va être tracée sur la table à tracer automatiquement comme la figure ci-dessous.

Figure 3:3:2:3 la procédure de tracer un courbe qui représent un label.

Donc, pour différencier les deux type façons de tracer, j’ai crée une variable dans le contexte globale, c’est : fileTypeHandling (‘csv’/ ‘lom’), donc quand j’ai choisi de afficher les paramètres dans un fichier, on va aussi analyser le type de fichier choisi, est modifier le type de fichier en cours de traitement dans le contexte globale via médiateur d’analyseur. Grâce à cette variable dans le contexte globale. À chaque fois on met un paramettre dans une table à tracer. Le function qui déclenche au dérrière va analyser la variable ‘fileTypeHandling’ pour vérifier quelle procédure il va éffectuer pour tracer le courbe.

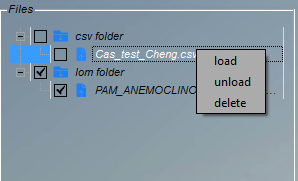
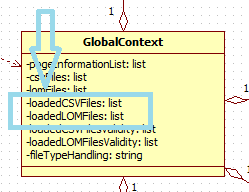
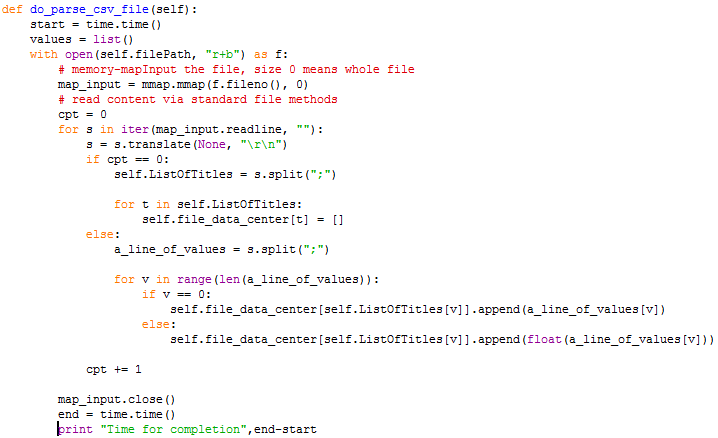
Comme nous le savons, dans la parie 3.3.1, on a modélisé un architecture que à chaque fois l’utilisateur ajouter une table à tracer sous une page, on va aussi lui attribuer une éspace ‘ plotting context’ dans l’environnement de travail. Le plotting context est pour sauvgarder toutes les informations nécessaires pour éffectuer les tracements. Et deuxième problème arrive, comment la table à tracer trouver les valeurs qui correspondant au paramètre mettre dedans. Il faut lui dire que les fichiers qu’il va aller chercher, donc, dans l’arbre des fichiers. Il faut crée une option de cliquer droit sur un fichier afin de charger un fichier dans le context globale.

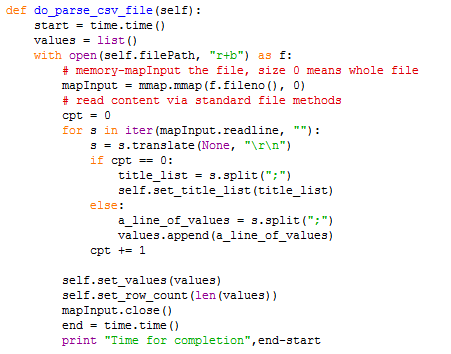
Figure 3:3:2:3 Charger un fichier dans le contexte globale

Donc deux list sont crée dans le contexte globale afin de stocker les fichiers chargé. Donc le table à tracer va savoir aller où pour chercher les valeurs du paramètre mis dedans.

Dans cette partie, j’ai aussi rencontré une difficulté. Souvent, quand on fait lecture d’un fichier csv, comme la figure 3:0, il y a beaucoup de manière grâce à les paramètres sont sauvgardé cologne par cologne. Dans un premier temps, je voudrais de stocker les valeurs dans le fichier via un structure key-value, chaque nom du paramètre est une clé pour obtenir la list de valeurs. Comme le code ci-dessous. Je pense que un structure de donnée de ‘key-value’ est va est facile à utiliser après.



Mais après avoir tester un fichier csv de taille 87M. Ça fait 6.5 second pour finir la lecture du fichier. Je trouve que ce n’est pas bien de laisser un utilisateur d’attendre longtemps.

Pour réduire le temps de lecture j’ai pensé à faire des lectures parallele avec des mutilthreading parce que théoriquement, coupé le fichier un nombre de partie selons le nombre de core de notre ordinateur, et faire des lectures parallele, c’est la plus rapide façon de lecture d’un fichier, mais après avoir analysé sur les Python, j’ai trouvé que l’éxecution de code python est contrôller par le ‘Python Vitual Machine’ (on l’appelle aussi : ‘interpreter main loop’). Python a été conçu de manière à ce que seulement un thread puisse être exécuté dans cette boucle principale. Et accès dans Python Vitual Machine est controller par le GIL (global interpreter lock) qui est là pour surveiller qu’il existe juste un thread qui exécute dans le PVM. Donc je n’ai pas passé beaucoup de temps à trouvé les solutions, mais utilisé une manière plus facile, c’est que je sauvgarde premier ligne comme la list des noms du paramètre. Et puis, le reste des valeurs, je les stocke dans un tableau en deux dimensions. De cette manière, le temps de lecture réduit beaucoup.



Pour récupérer la liste des valeurs d’un paramètre, il faut d’abord récupérer l’index du paramètre dans la list des noms du paramètre. Puis utiliser l’index trouvé pour parcourir la list des valeurs afin de produire une list pour le paramètre

### Gestion des actions utilisateur.

Dernière aspect qui est indispensable dans ma conception, c’est une espace qui sauvgarde tous les actions d’utilisateur, ça veut dire que à chaque fois, utilisateur réalise une action sur l’analyseur, alors un command, qui représent cet action, va saugarder automatiquement dans l’espace des actions dans le contexte globale. Les actions sont stocké sequentiellement dans une list de la classe ‘AnalyserAction’, ce module servira à restaurer une session de travail et réalisé une ‘undo’ action.

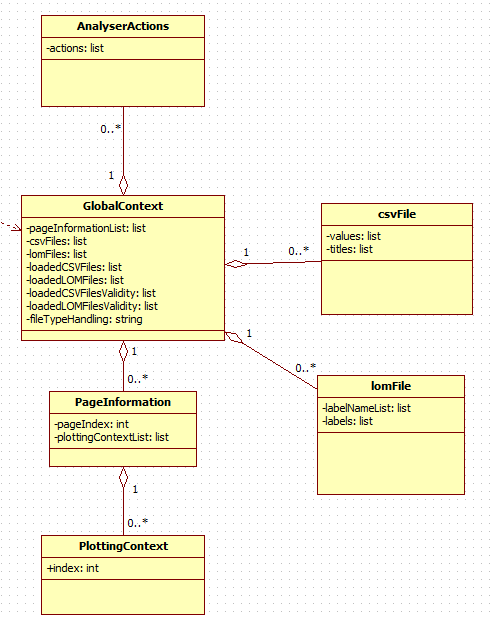
Jusqu’à ce moment là, on a finit de modéliser la partie modèle (je l’appelle: ‘contexte globale’ dans mon outil) complèt dans pattern Modèle-Vue-Contrôleur.

Figure 3:3:3:0 Charger un fichier dans le contexte global

Principe de fonctionnement est comme la figure ci-dessous, après avoir réalisé une session de travail, dans le contexte globale, tous les actions sont stocké séquentiellement. Et en fin, si utilisateur ferme le outil. Un fichier, qui inclut tous ces contenus dans un fichier, va être généré dans le répertoire prédéfinis.

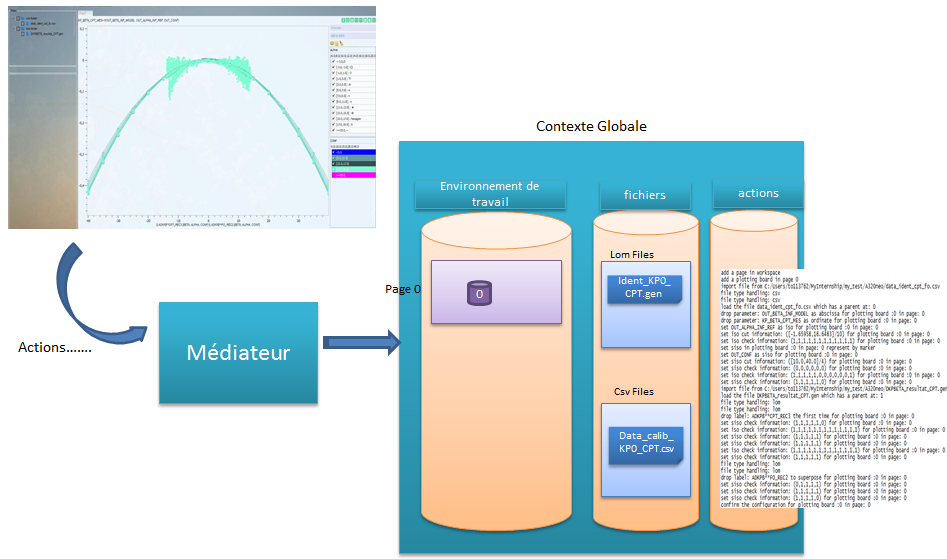


Figure 3:3:3:1 un cas test de l’environement de travail avec des actions

### Vérsion finale de modélisation.

Enfin on a obtenu une version finale de diagramme de classe qui représente mon architecture delogiciel. Les modules sont bien découpés en MVC, et je pense que cette architecture compose toutes les parties nécessaires qui correspondent à l’exigence de l’outil. Dans le chapitre 4, je vais réaliser étape par étape les focntionnalités demandé.

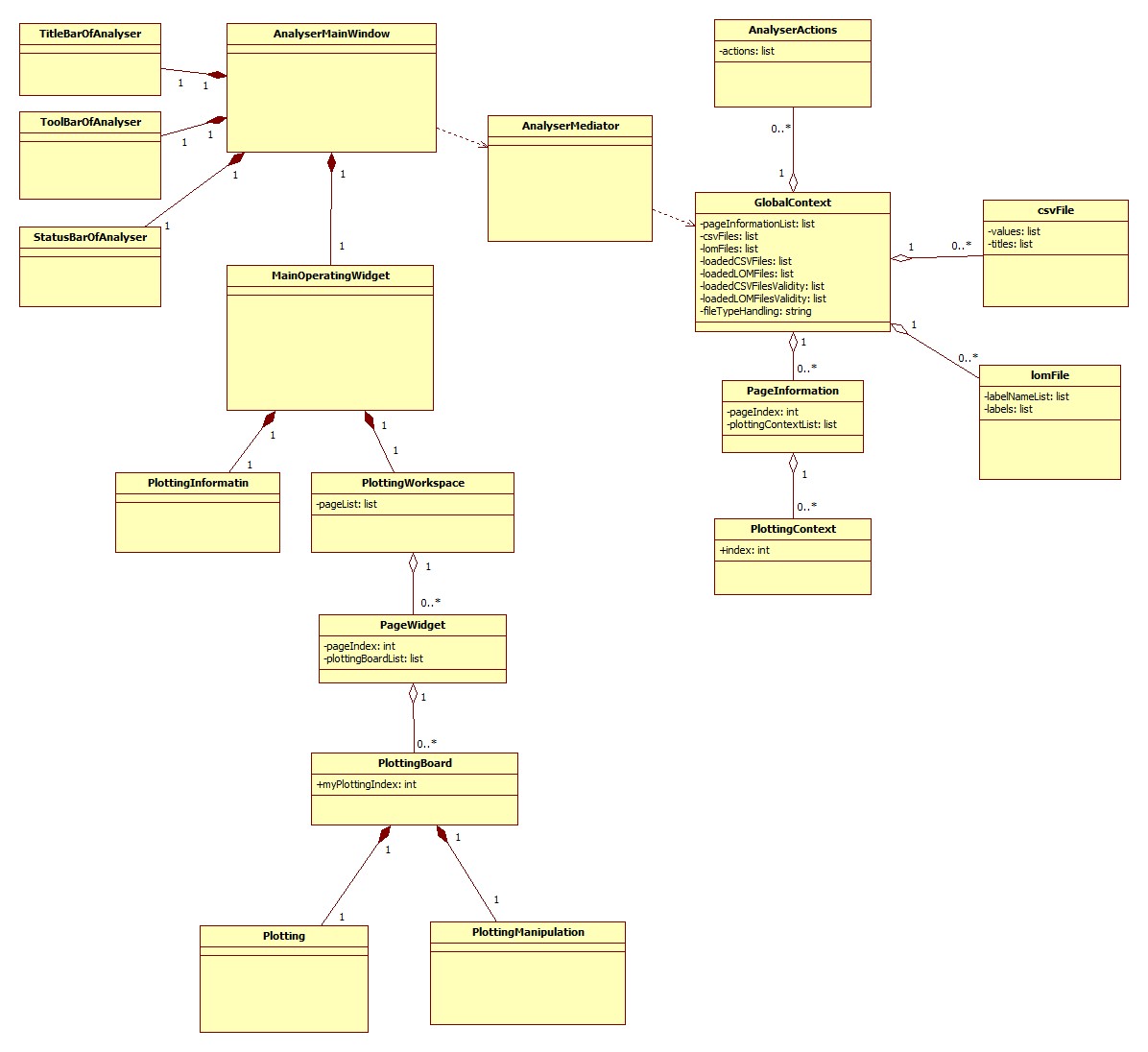


Figure 3:3:4 diagramme de classe de l’analyseur

# Fonctionnalités important

Après avoir finir de conçu l’architecture de l’analyseur, je commence à réaliser étape par étape les fonctionnalités demandé à réaliser. Dans la partie 3.3.1, j’ai expliqué que chaque fois on crée un table à tracer, il y a un plotting context qui va être attribué dans le contexte globale, le plotting context servir à sauvgarder les données nésséssaire pour faire tracer sur la table à tracer coorespondant.

Les données dans le ‘plottingContext’ sont sauvgardé sous un format comme la figure ci-dessous, un ‘key-value‘ architecture.

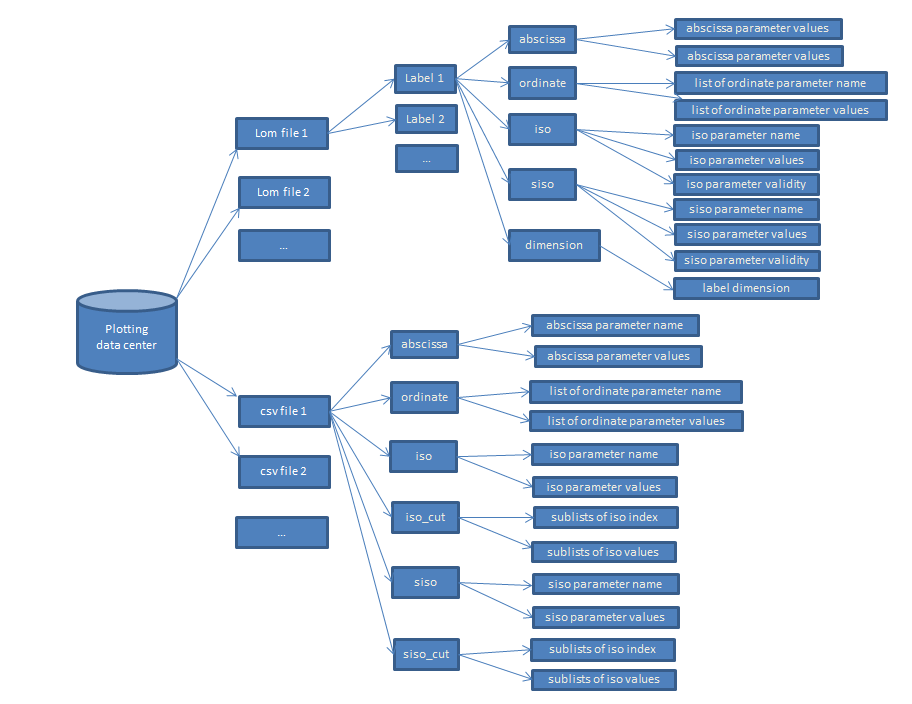
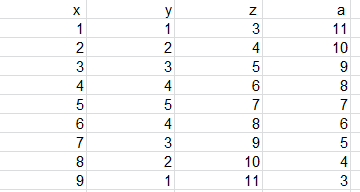


Figure 3:3:4 structure des données à tracer dans le ‘plotting contexte’

Dans la figure, on peut bien remarquer que les données à tracer sont retrouvé avec le nom du fichier, dans les données retrouvées avec le nom du fichier de type csv, on peut trouver un absciss, un iso, un super-iso et plusieurs ordinates possible. Et dans les données retrouvées avec le nom du fichier lom. On peut trouver des labels, et chaque label possède déjà une courbe bien configurée. Je vais détailler les procedures de tracement dans les paragraphs après successivement.

Pour bien expliquer les parties 4.1, 4.2, 4.3. Je vais d’abord crée un fichier de csv très simple qui est au même format avec des fichiers d’éssai.

## Ordinate = f (abscissa)

Maintenant, on va réaliser notre première fonctionnalité, c’est simplement tracer des courbes en deux dimensions, ça veut dire que just un courbe qui représent y = f(x). C’est une fonctionnalité que beaucoup de outils peuvent réaliser, même avec office excel. Mais c’est just une fonctionnalité de base, vous allez trouver beaucoup d’autre fonctionnalité après qui sont mes conceptions personnels.

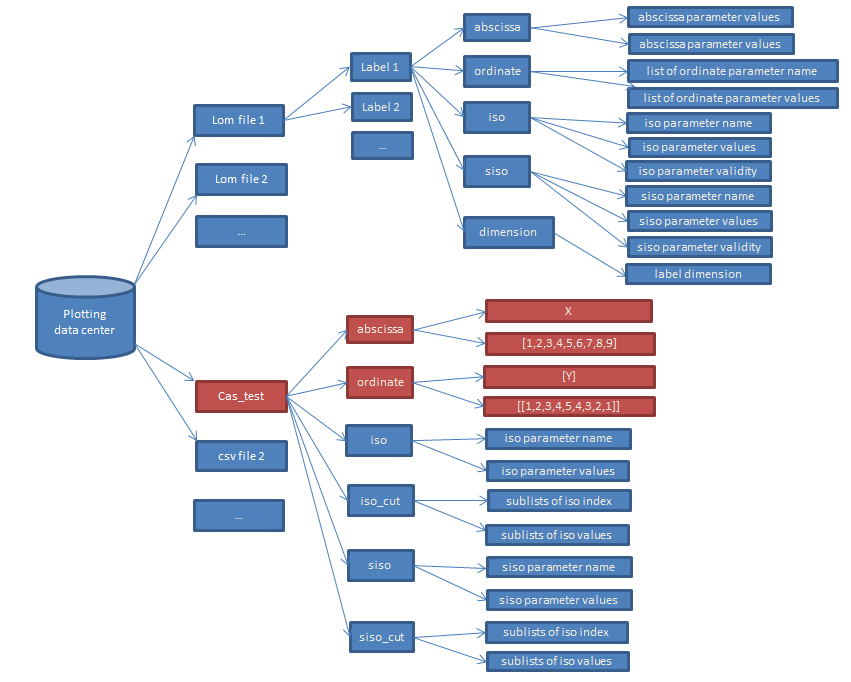
La procedure de tracement est déclenché par une ‘drag and drop’ action, c’est à dire que on prend un paramètre dans la list des paramètres à gauche, et on le mets sur la table à tracer, et puis une fois on relache le button gauche du souris, la fonction va être appelé. Et dans un premier temps, on va d’abord chercher la référence vers l’éspace de travil qui correspond à cette table à tracer via le numéro de la page et le numéro de la table à tracer sur cette page. Ensuite, si on est en traine de travailler sur des fichiers d’éssai de type csv, on va chercher la list des valeurs dans des fichiers sauvagardé dans le modèle et les stockers dans la structure des données à tracer dans le ‘plottting context’. Selons le nom du fichier et son ordonnée choisi par l’utilisateur.

Figure 4 :1 :0 l’abscisse et l’ordonnée d’une courbe définit

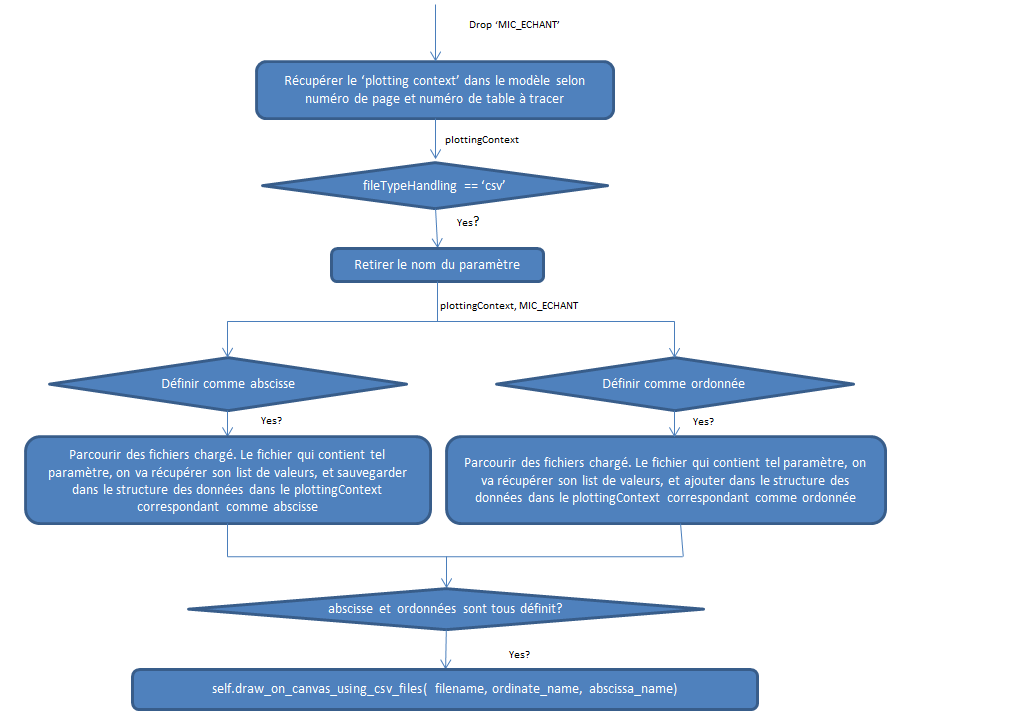
Et puis, une fois que l’abscisse et l’ordonnée sont été déjà configurées pour au minimum un fichier importé. Et alors une fonction de tracement va être appelée automatiuement afin de tracer une courbe sur la table à tracer. Alors la fonction va just récupérer tous les pairs de donnée (x, y) et construire autant nombre de courbe successivment différencier par une list de couleur prédéfinis.

Figure 4 :1 :1 la prodécure pour tracer des courbes y = f(x)

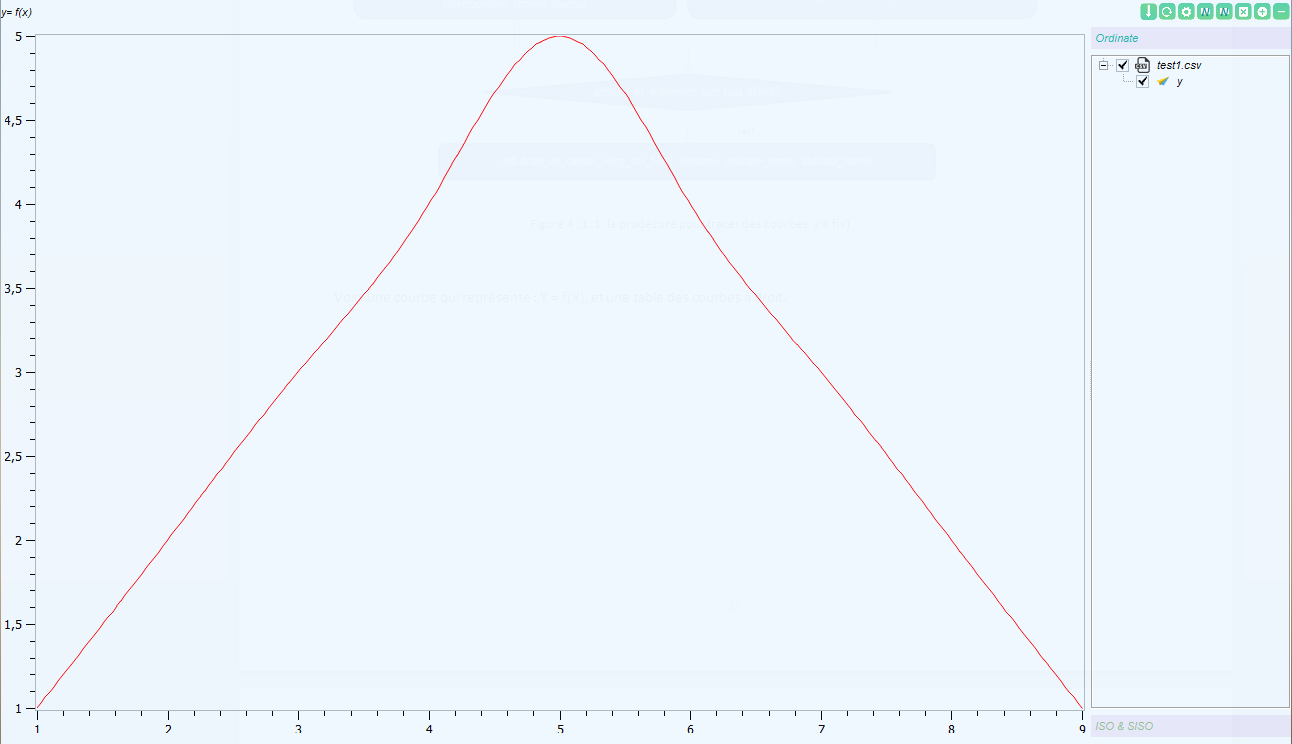
Voici une courbe qui représente : Y = f(X) et une table des courbes à droit.

Figure 4 :1 :2 une démonstration du tracement de ‘Y = f(X)’

Comme nous le savons, sur un table à tracer, on peut construire beaucoup de courbe, mais on peut seulement définir une seule abscisse car sinon, il n’y a pas de sense pour analyser. Dans ce cas, on peut facilement rétrouvé un courbe sur la table via son ordonnée et son fichier, donc, j’ai aussi crée une table de référence qui est rélié avec la table à tracer, c’est un ‘tree widget’, dedans, chaque paramètre ordonnée sous un fichier csv est relié avec un courbe sur la table. On peut faire dispparître et afficher d’une courbe via choisir et déchoisir l’option à droit. Dans la partie 4.4, on va détailler la fonctionnalité expiqué dans ce paragraph.

## Ordinate = f (abscissa, iso)

Dans la partie précédente, on a expliqué comment tracer les courbes en deux dimensions, mais si on veut virtualiser un troisième paramètre qui correspondant des points sur les courbes existants, normalement, on va les tracer sur un répère en 3 dimensions. Mais dans mon outil, je dois représenter cette troisième dimension (on va l’appeler ‘iso’ dans les paragraphs suivants) sur la courbe en deux dimensions. Ça veut dire que sur la courbe existant, on va d’abord couper l’iso par des intervalles, et puis, on attribue un marqueur ou un coleur pour chaque interval. Enfin on peut obtenir notre courbe qui représent par différent couleur ou marqueur. Pour réaliser de configurer un iso pour les courbe existant. On a trois étapes à réaliser.

Etape 1 : On prend le paramètre dans la table à gauche et le met dans ‘isoLabel’, et puis, la procédure dans le diagramme de séquence ci-dessous va être déclenché pour configurer les informations sur iso dans la context du table à tracer.

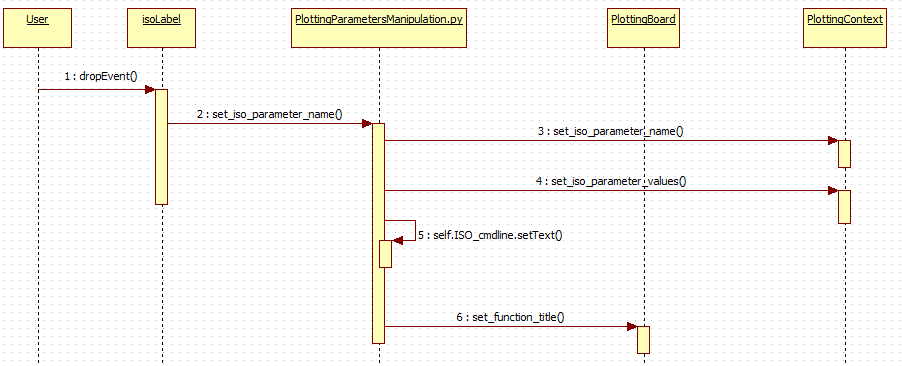


Figure 4:2:0 la prodécure pour configurer le iso pour une table à tracer

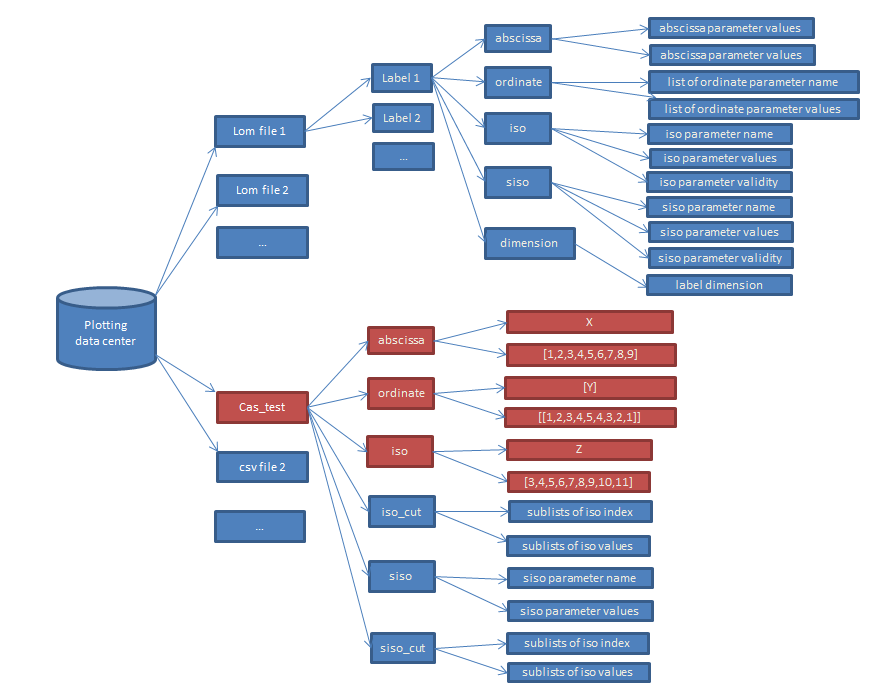
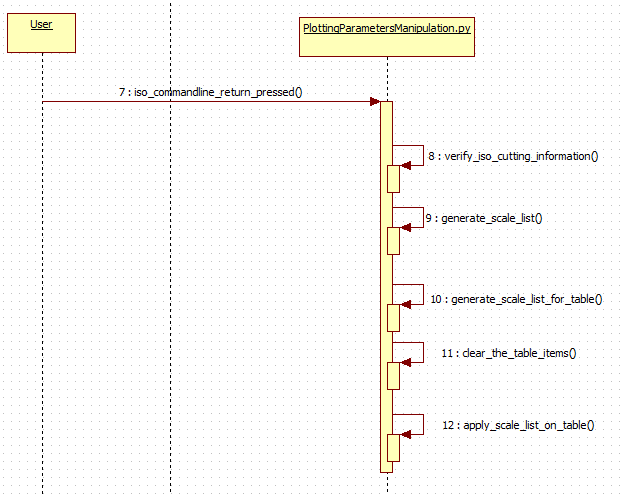
La figure ci-dessous vous montre ce qui c’est passé dans la ‘plotting context’ qui corresponds à cette table à tracer

Figure 4:2:1 l’abscisse, l’ordonnée et iso d’une courbe définit

Etape 2: On va tapez et confirmer un command pour découpe la list des valeurs de ‘Z’. Et puis générer tous les composants nécessaires pour les manipulations après. On va l’expliquer via un exemple.



‘Z’ = [3, 4, 5, 6, 7, 8, 9, 10, 11]  
 isoCutCommand = 4;6;8

Confirmer ? ==>

sub\_list\_iso\_value: (4, '<5') : [3,4]

(6, '[5,7['): [5,6]

(8, '>=7') : [7, 8, 9, 10, 11]

sub\_list\_iso\_index: (4, '<5') : [0, 1]

(6, '[5,7['): [2, 3]

(8, '>=7') : [4, 5, 6, 7, 8]

==> attribué deux list dans la ‘plotting context’ selons le nom du fichier csv.

==>générer des ‘combo box’ pour tous les trois intervalles et les Mettre sur la table pour iso

Figure 4 :2 :2 la prodécure pour manipuler le iso pour une table à tracer

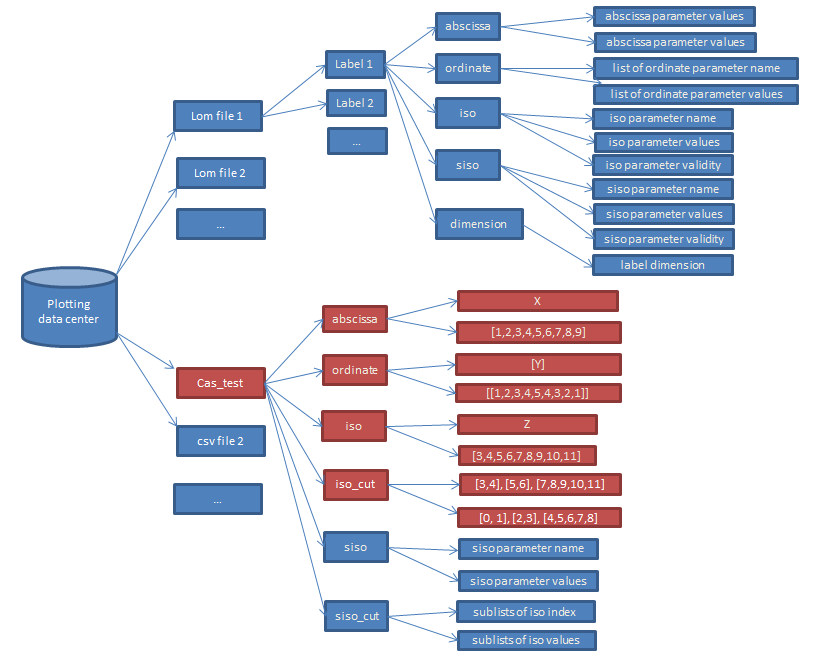
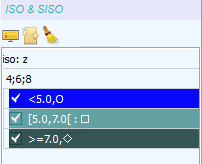


Figure 4 :2 :3 l’abscisse, l’ordonnée, l’iso et l’iso coupé d’une courbe définit

Etape 3, une fois que l’utilisateur choisit les boites options, une procédure comme le digramme de séquence ci-dessous va faire renouveller l’ensemble des courbes exitant. L’idée c’est que pendant l’exécution de la méthode ‘cut\_current\_curve\_by\_iso’. On va analyser l’iso coupé afin de réconstruire autant nombre de courbe. Par exemple, si utilisateur choisit tous les trois intervalles, on va d’abord éffacer la courbe qu’il existe déjà et réconstruire trois courbe pour chaque intervalle de l’iso en configurant les couleurs ou marqueurs correspondants.

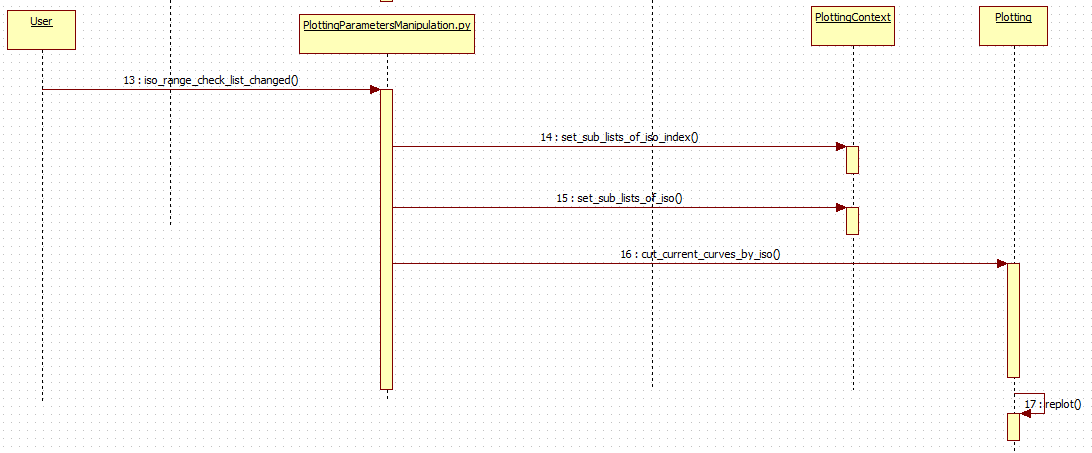


Figure 4 :2 :4 la prodécure pour mettre en œuvre le découpage sur iso pour une table à tracer

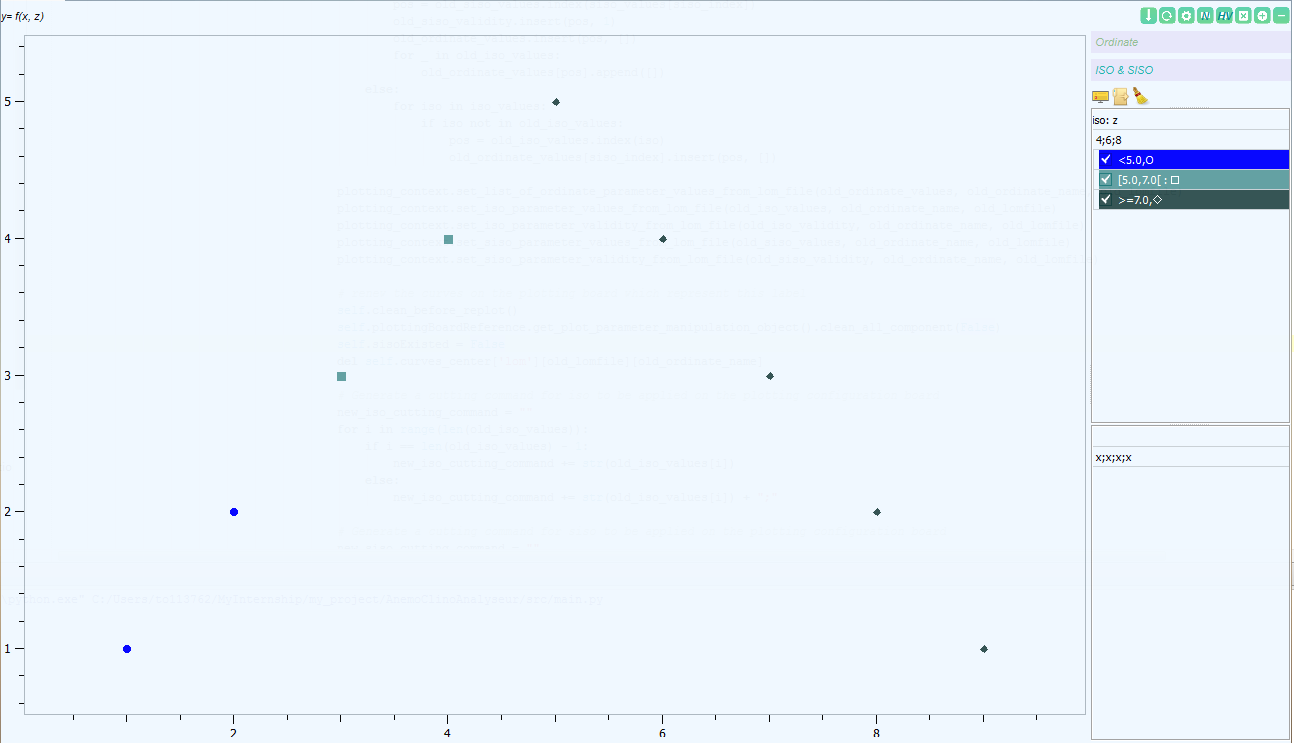
Voici le résultat obtenu : la courbe y = f(x) coupé par iso: 4;6;8

Figure 4 :2 :3 un courbe avec iso : y = f(x, z)

## Ordinate = f (abscissa, iso, siso)

Maintenant, on va parler sur comment ajouter une quatrième dimension (on va l’appelle : siso ou super-iso dans les paragraphs suivantes) sur le courbe existant. Pour réaliser de configurer un super-iso pour la courbe existant. On a aussi trois étapes à réaliser.

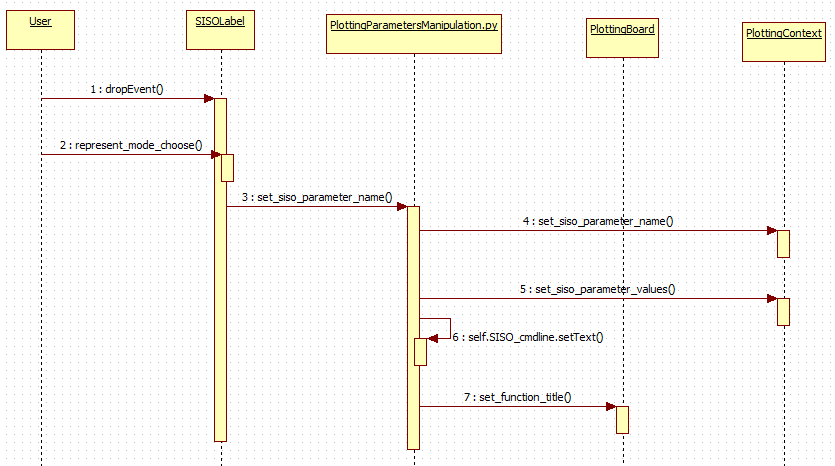
Etape 1 : On prend le paramètre dans la table à gauche et le met dans ‘sisoLabel’, et puis, la procédure dans le diagramme de séquence ci-dessous va être déclenché pour configurer les informations sur super-iso dans la context du table à tracer. première étape c’est presque pareil que la configuration de l’iso. Sauf que maintenant, on va choisir son mode de représentation pour bien identifier iso et siso. Donc soit on représent siso par couleur et alors représent iso par marqeur.

Figure 4:3:0 la prodécure pour configurer le siso pour une table à tracer

La figure ci-dessous vous montre ce qui c’est passé dans la ‘plotting context’ qui corresponds à cette table à tracer, donc on a configuré un super-iso pour la courbe existant en sauvgradant le nom du paramètre et la list des valeurs dans la ‘plotting context’

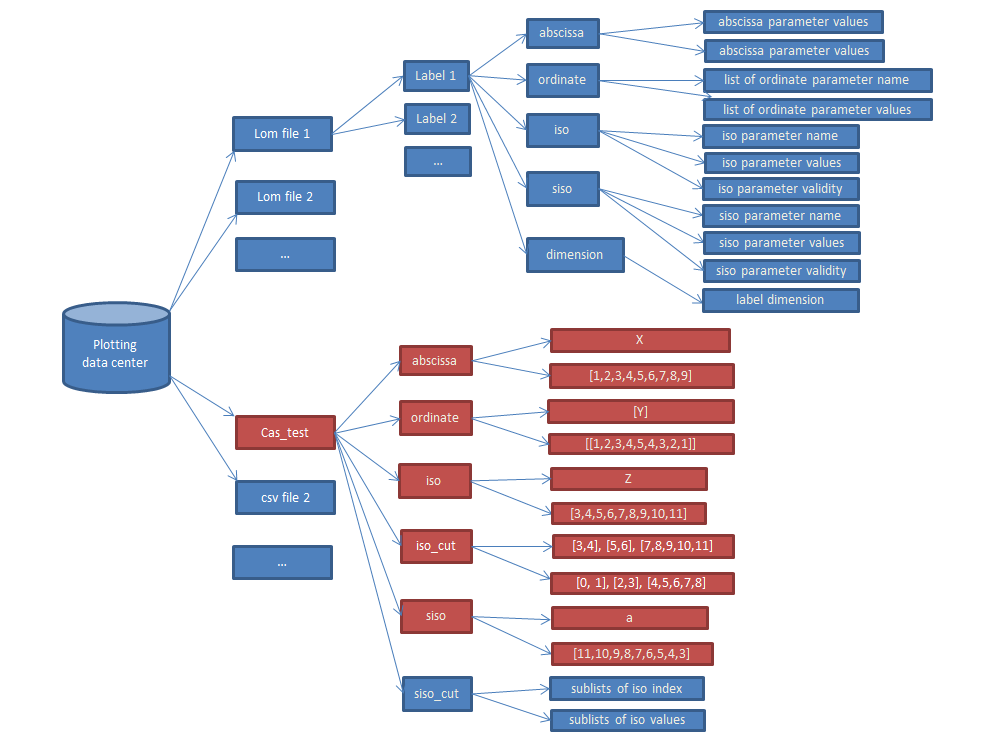
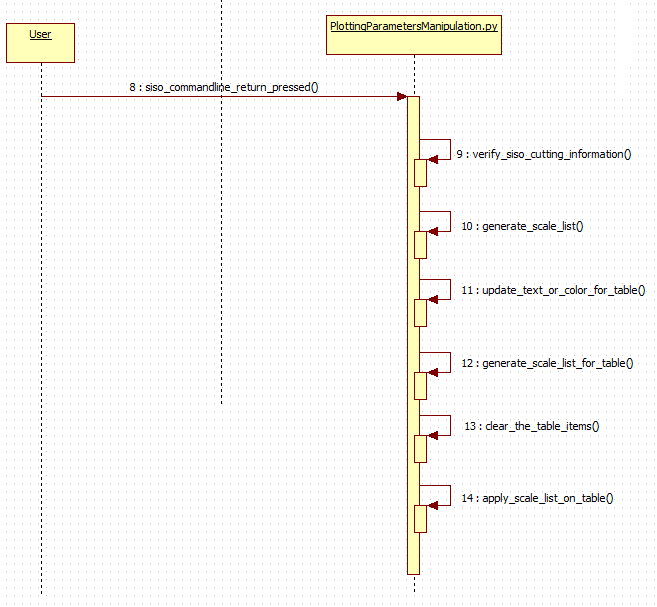


Figure 4:3:1 l’abscisse, l’ordonnée, l’iso et le super-iso d’une courbe définit

Etape 2: On va tapez et confirmer un command pour découpe la list des valeurs de ‘Z’. Et puis générer tous les composants nécessaires pour les manipulations après. On va l’expliquer via un exemple.



‘a’ = [11, 10, 9, 8, 7, 6, 5, 4, 3]  
 isoCutCommand = 5;7

Confirmer ? ==>

sub\_list\_iso\_value: (5, '<6') : [5, 4, 3]

(7, '>=6'): [11, 10, 9, 8, 7, 6]

sub\_list\_iso\_index: (5, '<6') : [6, 7, 8]

(7, '>=6'): [0, 1, 2, 3, 4, 5]

==> attribué deux list dans la ‘plotting context’ selons le nom du fichier csv.

==>générer des ‘combo box’ pour tous les trois intervalles et les Mettre sur la table pour spuer- iso

Figure 4 :3 :2 la prodécure pour manipuler le siso pour une table à tracer

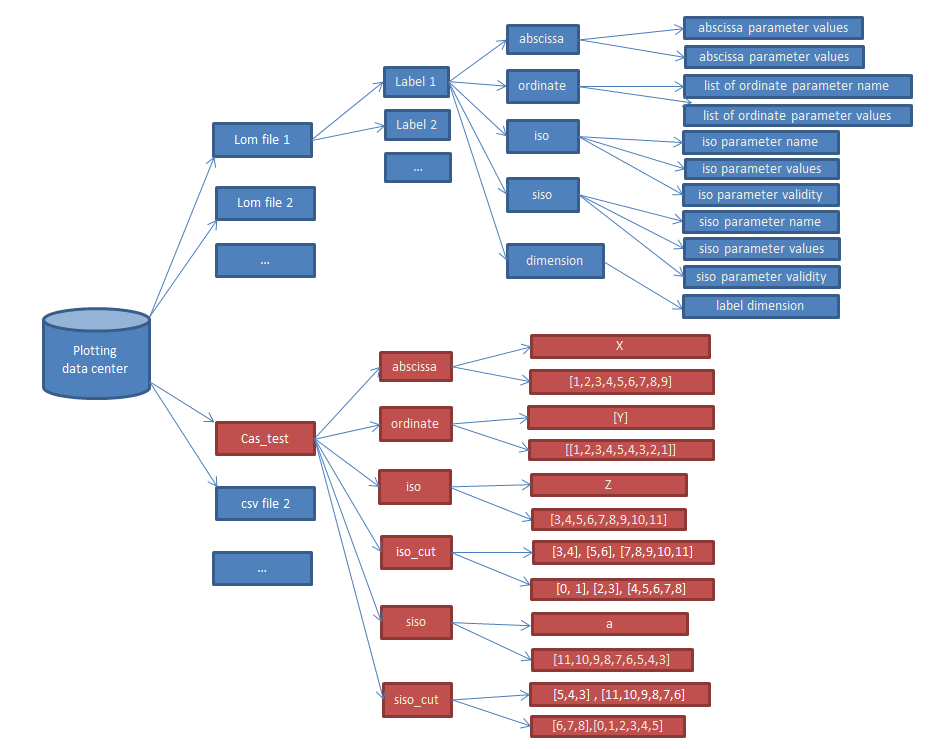
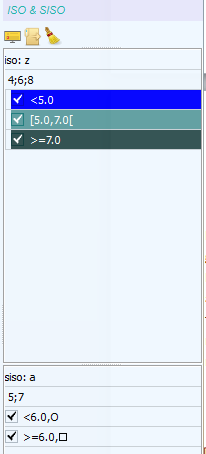


Figure 4 :3 :3 l’abscisse, l’ordonnée, l’iso, l’iso coupé, le super-iso et le super-iso coupé d’une courbe définit

Etape 3, une fois que l’utilisateur choisit les boites options, une procédure comme le digramme de séquence ci-dessous va faire renouveller l’ensemble des courbes exitant. L’idée c’est que pendant l’exécution de la méthode ‘cut\_current\_curve\_by\_siso’. Comme on a choisit de représenter le super-iso par marqueur, on va analyser l’siso coupé, et puis pour chaque courbe déjà coupé par les intervalles de iso coupé, on va analyser successivement les intervalles de super-iso coupé afin de configuré le bonne marqueur pour le point selons son indice.

Par exemple, si utilisateur choisit tous les trois intervalles, on va d’abord éffacer la courbe qu’il existe déjà et réconstruire trois courbe pour chaque intervalle de l’iso en configurant les couleurs ou marqueurs correspondants.

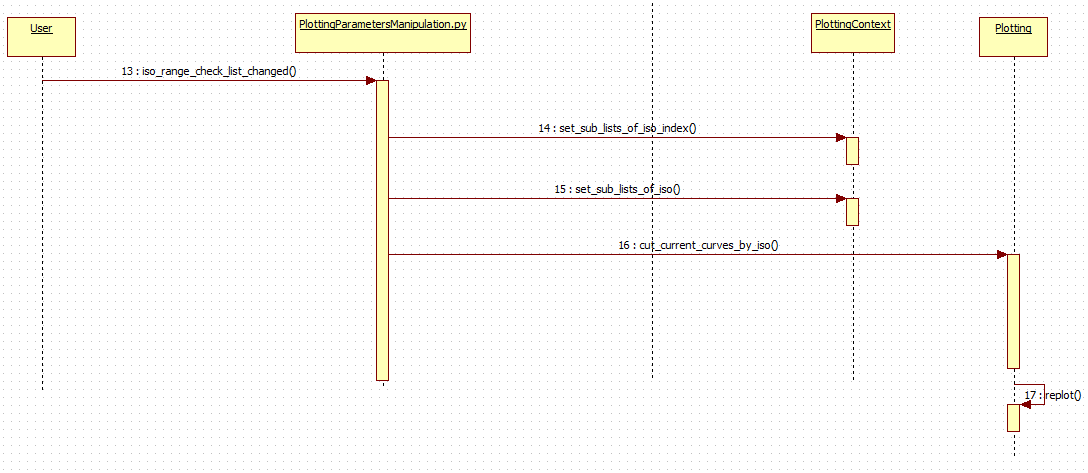


Figure 4 :2 :4 la prodécure pour mettre en œuvre le découpage sur siso pour une table à tracer

En fin, on obtient un courbe y = f(x) qui a été déjà coupé avec l’iso : ‘z’ et le super-iso : ‘a’.

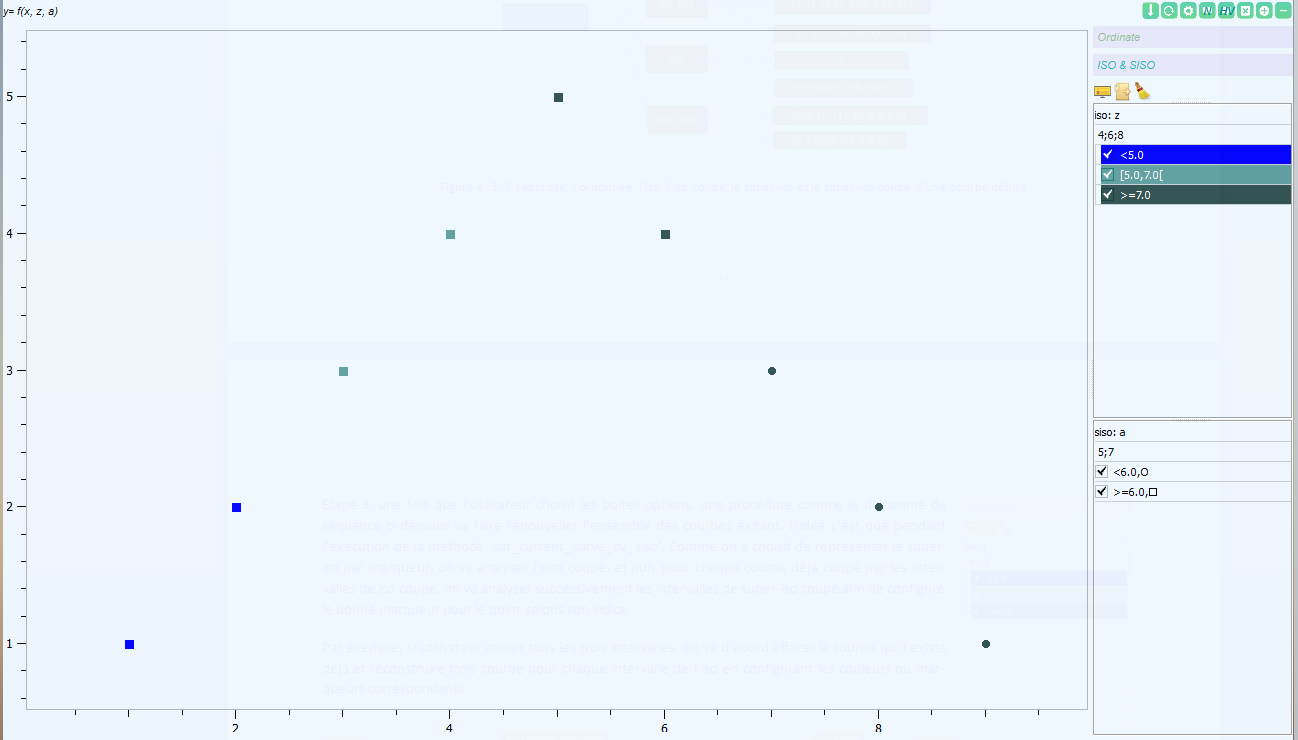


Figure 4 :2 :3 un courbe avec iso : y = f(x, z, a)

## Tracement d’un label

Un label est l’unité de base d’un fichier de la loi de command. Un label contient tous les informations. La lecture du fichier de la loi de command est réalisée par un module qui a été déjà développé dans l’équipe.

La procédure du tracement d’un label est pareille que tracer une courbe qui est composé par des paramètres dans les fichiers d’éssai. Mais si on choisi le fichier de LOM à travailler avec, le variable : fileTypeHandling va changer automatiquement en ‘lom’. Et puis une fois que l’utilisateur prend un label dans la list de label à gauche et le met dans une table à tracer, une procédure comme la figure ci-dessous va être déclencé pour récupérer les informations sur des ordonnéés, les sauvgarder dans la context et enfin réaliser le tracement du label sur la table à tracer.

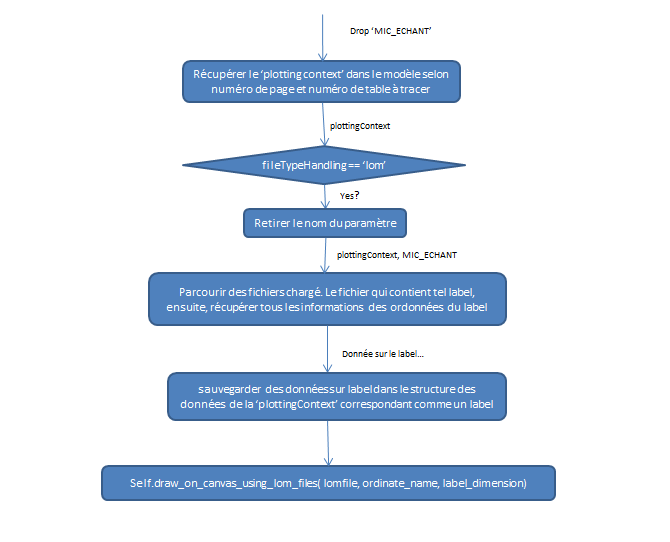
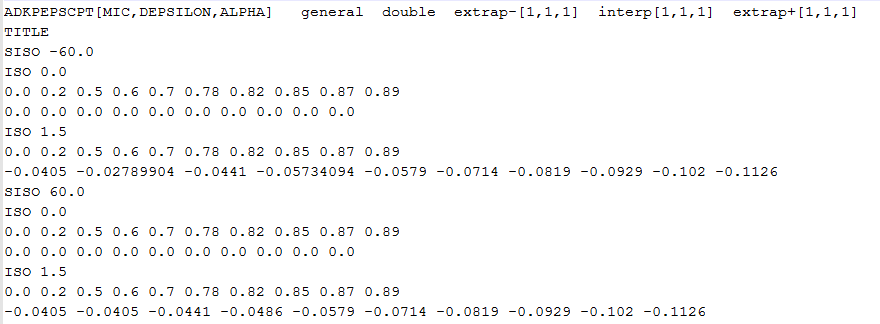


Figure 4 :4 :0 la prodécure pour tracer des courbes qui représent un label

Les données sur un label vont sauvgarder dans une structure des données comme le format ci-dessous.

data\_center\_for\_the\_file={'abscissa':[abscissa\_parameter\_name,abscissa\_parameter\_values],  
 'ordinate':[list\_of\_ordinate\_parameter\_name,list\_of\_ordinate\_parameter\_values],  
 'iso':[iso\_parameter\_name,iso\_parameter\_values,iso\_parameter\_validity],  
 'siso':[siso\_parameter\_name,siso\_parameter\_values,siso\_parameter\_validity],  
 'dimension':x}  
  
self.data\_center[lom\_file\_name][label\_name] = data\_center\_for\_the\_file

Une fois que les données sont bien sauvgardé. On peut facilement produire l’ensemble des courbes. Maintenant, on va analyser un cas de test sur le tracement d’un label en dimension 3 dans un fichier de la loi de command.

.

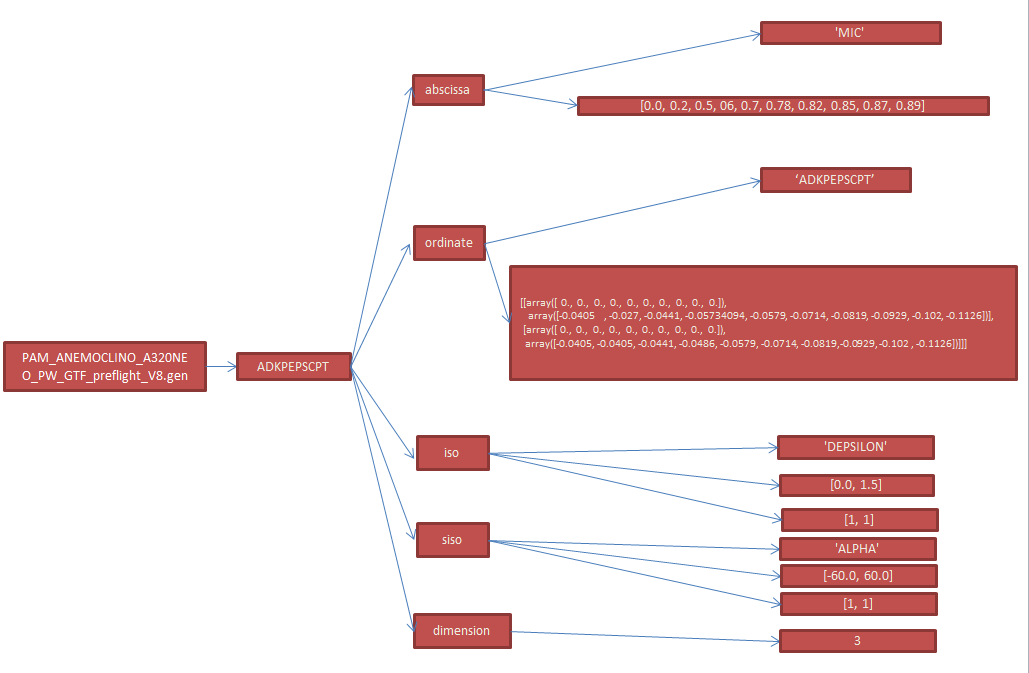
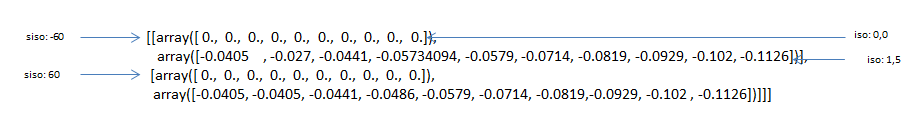
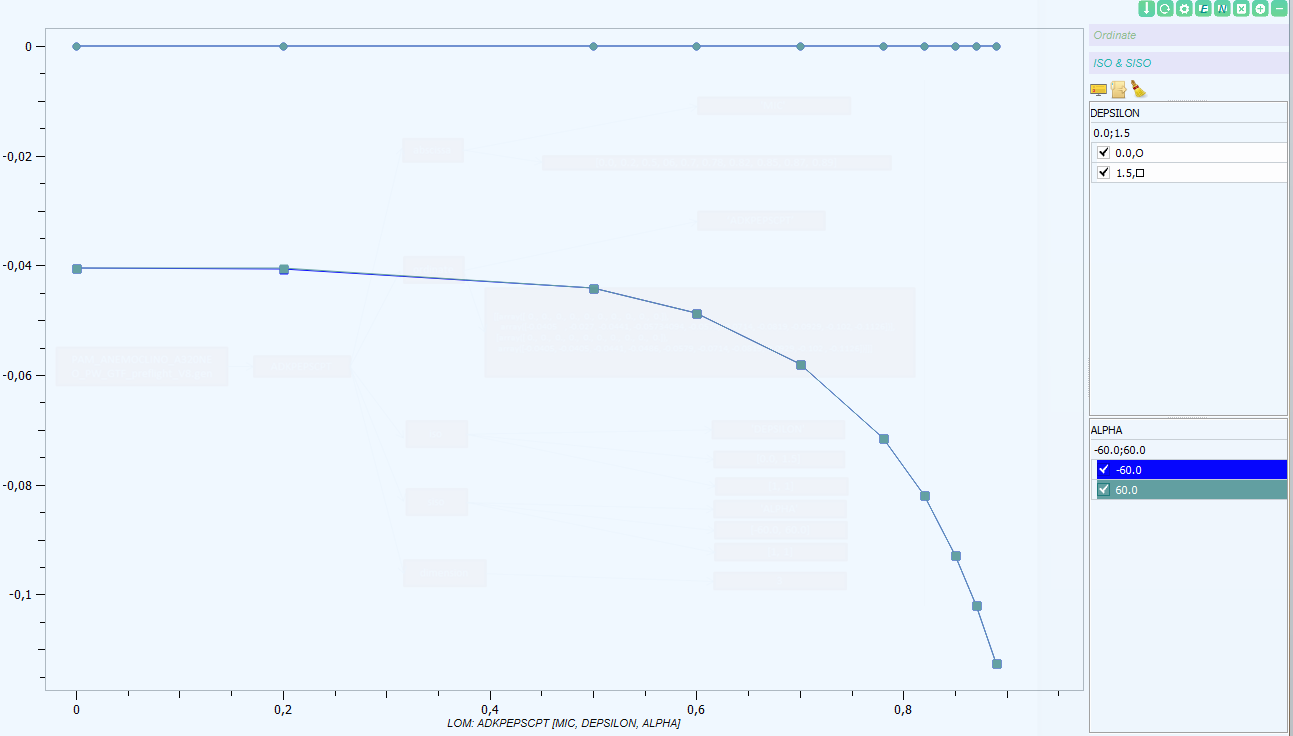
Si on met le label sur une table à tracer, la méthode déclenché va rempli les informations sur ce label dans le base de donnée de son context comme la figure ci-dessous.

Figure 4 :3 :3 les informations sur label ADKPEPSCPT sauvgardé dans le context

Vue que c’est un label en trois dimension, ça veut dire que il existe iso et super-iso. Donc les données sur l’axe ordonné est un tableau en deux dimension comme le format ci-dessous.



Voici le résultat obtenu:

## Labels Superposition

Dans la partie précedente. On a déjà parlé sur comment tracer un label sur une table, mais en réalité, il existe aussi des cas que l’utilisateur veut mettre plusieurs label sur a même table. C’est-à-dire une superposition des labels. Donc, Si utilisateur importé un deuxième label dans la table à tracer, on va vérifier si le label viens d’importer a la même dimension que le label existe déjà. Si oui, on va passer dans la procédure de superposition, sinon, on va laisser utilisateur choisir s’il veut écraser le label existe ou abandonner cet action

Drop ADKPEPSFO

Récupérer le ‘plotting context’ dans le modèle selon numéro de page et numéro de table à tracer

PlottingContext

FileTypeHandling == ‘lom’

Yes?

Retirer le nom du paramètre

PlottingContext, ADKPEPSFO

Boucle finis ?

Prendre un lomfile dans la loaded\_lomfiles

Lomfile

No

ADKPEPSFO existe dans le lomfile ?

Yes

La table à tracer existe dèjà un l’ensemble des courbes pour un label ?

No

Yes

No

La dimension du label existe est différent que le label vient de importé ?

Yes

No

Utilisateur actualiser la table à tracer avec nouveau label, éffacer ancien label

Yes

Effacer la table à tracer

Sauvegarder des données sur label dans le structure des données de la ‘plottingContext’ correspondant comme un label

La procédure de superposition

Self.draw\_on\_canvas\_using\_lom\_files( lomfile, ordinate\_name, label\_dimension)

`

Pour la procédure de superposition : on va tout d’abord parcourir les anciens labels tracés sur la table afin de vérifier si le label qui vient d’importé contiens les informations de découpage différent. Si oui, on va faire union avec la nouvelle information de découpage. Puis actualiser les informations de découpage de ces auciens labels et ses valeurs d’ordonnée. Enfin, on va éffacer la table à tracer et puis retracer l’ensemble des courbes avec son nouveau information de découpage.

Prendre un label dans la list des labels déjà tracé

Dim = 3

Dim = 2

Dim = 1

Faire Union sur les informations des iso ou siso pour ce label

Faire Union sur les informations des iso pour ce label

Actualiser le tableau en deux dimensions sur le label existant

Actualiser le tableau sur le label existant

Effacer et retracer ce label

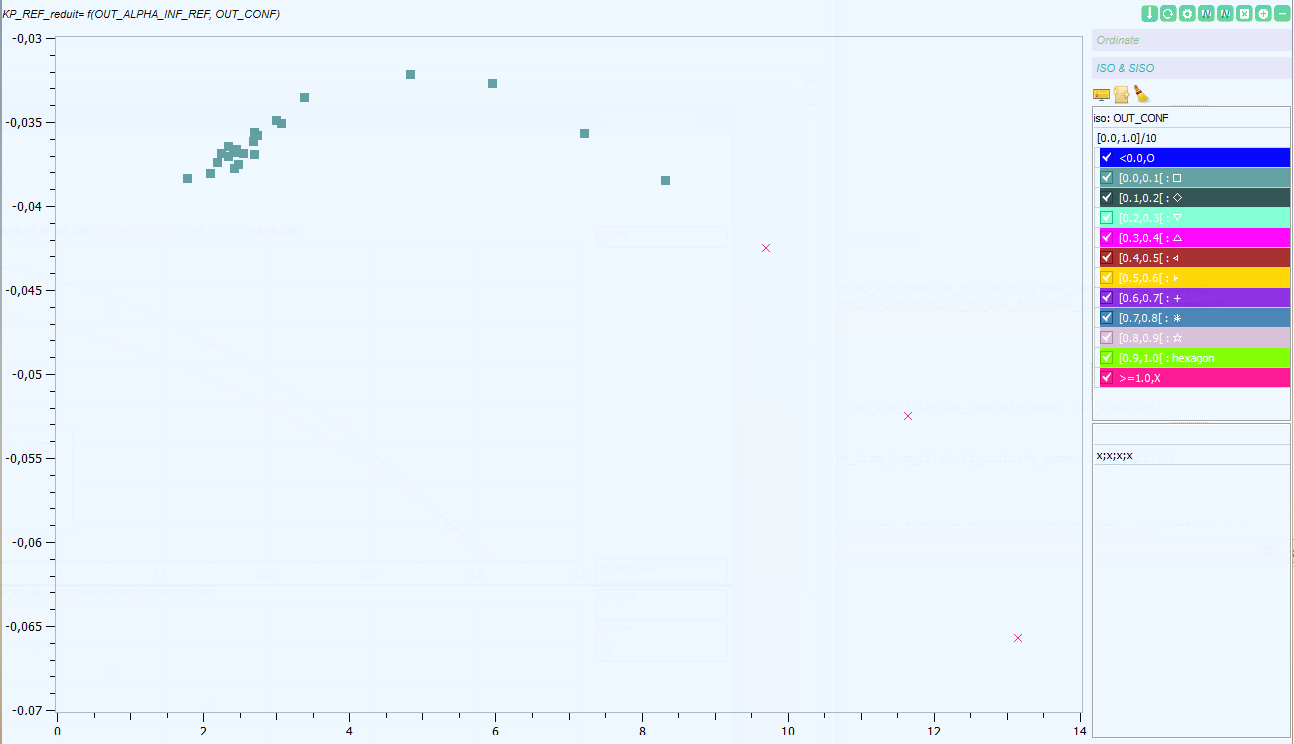
Effacer et retracer ce label

Actualiser les informations de découpage pour le label qui viens d’importé

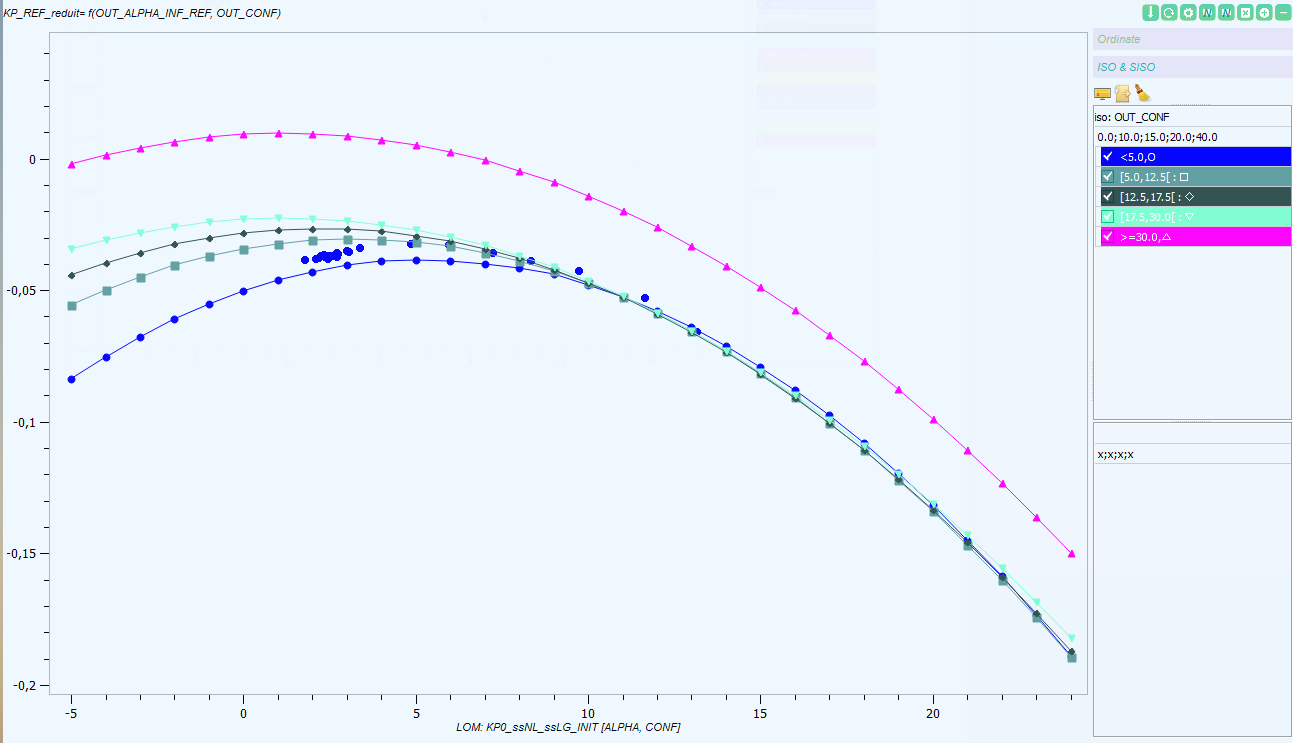
Tracer le nouveau label qui viens d’importé

## Superposition des Labels sur des courbes d’éssai

Après avoir réalisé le tracement des labels qui viennent du fichier de la loi anémométrique. On veut maintenant superposer des labels sur un ensemble des courbes qui a été tracé par les paramètres dans les fichiers d’éssai. Pour cela, on va d’abord vérifier si la dimension du label importé est pareil que la dimension de l’ensemble des courbes qui viennent des fichiers d’éssai. Si oui. On va remplacer les informations de découpage et actualiser l’ensemble des courbes avec nouveau information de découpage. Et enfin, on va tracer le label importé directement.

Voici un cas de test. On a un ensemble des courbes tracé par des valeurs dans un fichier d’éssai qui représent : KP\_REF\_reduit = f (OUT\_ALPHA\_INF\_REF, OUT\_CONF). Donc la dimension est 2.

Ensuite, on va importer un label : KP0\_ssNL\_ssLG\_INIT qui a aussi une dimension de 2. Donc vous trouvez en figure ci-dessous. On a déjà actualisé l’ensemble des courbes qui viennent du fichier d’éssai avec les nouvelles informations de découpage, et on a aussi l’emsemble des courbes qui représent le label :



## Manipulation des courbes

Selons les six parties précédentes. On peut trouver dans la structure des données

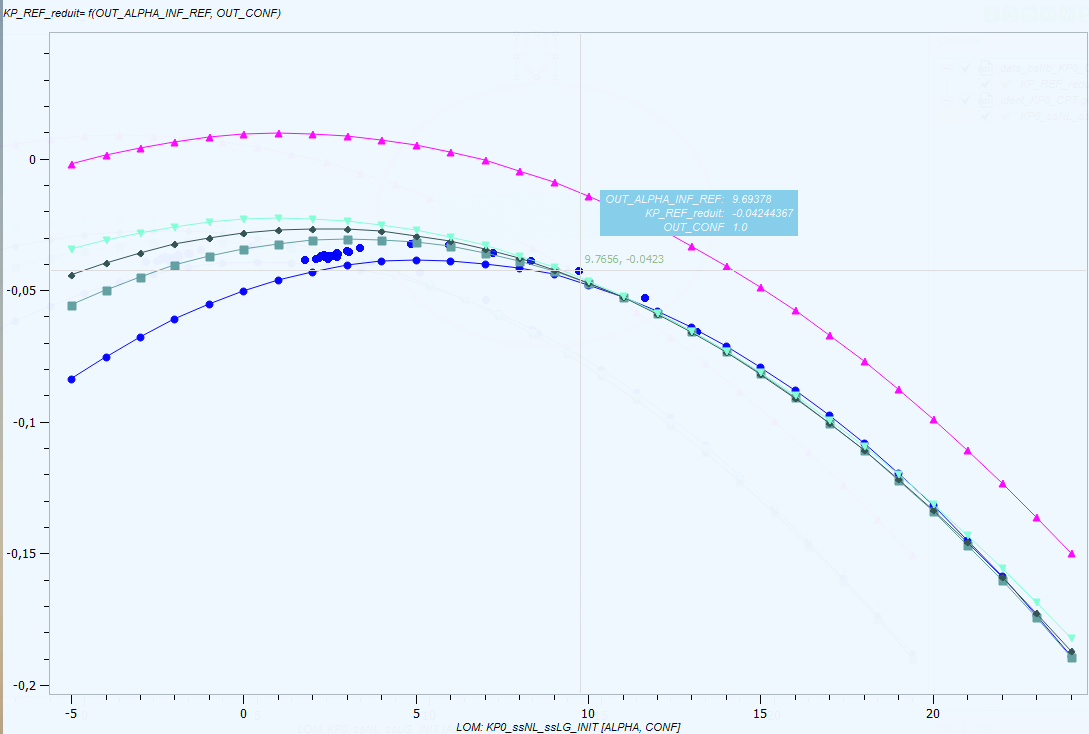
## Suivi & Affichage des points sur graphe

Pour réaliser le suivi et l’affichage d’un point qui est sur le graphe quand le pointeur de souris s’arrête sur le point ou autour du point dans une région toléré. Tout d’abord, il faut avoir une mecanisme pour assurer que le mouvement du pointeur de la souris sur le graphe va déclencher une function pour suivre et afficher le point trouvé. Et enfait, je n’ai pas trouvé des signals par défaut dans le composant Qwt.QwtPlot, donc j’ai crée une classe de fils ‘ObjectEventService’ qui hérite de QObject pour la classe de Qwt.QwtPlot. et dans ce classe de fils.il existe des signals nécéssaires pour utiliser après, surtout, dans ce sujet, on a le signal qui représent le mouvement du pointeur de souris.

class ObjectEventService(QObject):  
 def \_\_init\_\_(self, parent):  
 QObject.\_\_init\_\_(self, parent)  
 parent.setMouseTracking(True)  
 parent.installEventFilter(self)  
  
 def eventFilter(self, \_, event):  
 if event.type() == QEvent.MouseMove:  
 self.emit(SIGNAL("MouseMove"), event.pos())  
 elif event.type() == QEvent.MouseButtonPress:  
 self.emit(SIGNAL("MouseClicked"), event.pos())  
 elif event.type() == QEvent.MouseButtonRelease:  
 self.emit(SIGNAL("MouseReleased"), event.pos())  
 elif event.type() == QEvent.MouseButtonDblClick:  
 self.emit(SIGNAL("MouseDoubleClick"), event.pos())  
 elif event.type() == QEvent.HoverEnter:  
 self.emit(SIGNAL("MouseHovered"), event.pos())  
 elif event.type() == QEvent.Leave:  
 self.emit(SIGNAL("MouseLeaved"))  
 return False

Donc il est suifi d’instancier un objet de fils de type ‘ObjectEventService’ qui hérite le canvas dans le Plotting objet, et puis, on va utiliser une function connect de crée le signal/slot mechanisme entre le graphe et la méthode que l’on va appeler.

self.canvas = ObjectEventService(self.plotting.canvas())  
self.connect(self.canvas, SIGNAL("MouseMove"), self.point\_tracking\_procedure)



La procédure s’analyse le point dans le fichier d’éssai ou de la loi d’anémométrique comme le digramme ci-dessous :

(abscissa\_point, ordinate\_point)

OperationMode == 1

Yes

Récupérer le ‘plotting context’ dans le modèle selon numéro de page et numéro de table à tracer

**data\_center** = plotting\_context.get\_data\_center ()

curves\_on\_canvas : Les états d’affichage des courbes

Boucle terminé ?

**Prends une clé primaire, plutôt un nom du fichier, dans data\_center**.keys ()

filename

‘gen’

‘csv’

**Prends un ordonée dans data\_center**[**filename**]

**Prends un label dans data\_center**[**filename**]

**Prendre des valeurs d’abscisse et des ordonnées afin de calculer le rayon horizontal: self.radius\_x et le rayon vertical : self.radius\_y**

**Prendre des valeurs d’abscisse et des ordonnées afin de calculer le rayon horizontal: self.radius\_x et le rayon vertical : self.radius\_y**

**D’une autre manière**

**Parcourir les parirs de list (abscisse\_valeurs, ordonnée\_valeurs) afin de trouvée des points qui est dans la région toléré**

**Parcourir les parirs de list (abscisse\_valeurs, ordonnée\_valeurs) afin de trouvée des points qui est dans la région toléré**

No

No

Un Point trouvé ?

Un Point trouvé ?

**Composé des informations d’affichages**

Yes

**Composé des informations d’affichages, si l’utilisateur a configuré des paramètres supplémentaires à afficher, il est suiffit de juste utiliser l’indice du point trouvé afin de trouvé le valeur correspondant.**

Yes

**Afficher les informations comoposés**

## Edition des courbes tracé par les labels

On va parler maintenant comment modifié le un label d’un fichier de la loi anémométrique.

Pour cela, on doit tout d’abord entrer dans le mode ‘édition des labels’ (self.canvasOperationMode == 2) en configurant le label à modifier, et ensuite, on a besoin de capturer le moment quand on click le button gauche de la souris car ça va déclencher la procédure de recherche, si le coordonnée où le pointeur se positionne sur le graphe est dans la région toléré d’une point dans l’ensemble des courbes de label. Donc, dès ce moment là, si on bouge le souris sur le graphe, la valeur d’ordonnée du point trouvé va être modifié selons le l’ordonnée du point où le pointeur du souris se positionner sur le graphe, et en même temps, l’ensemble des courbes qui représent le label va être actualisé avec nouveau contenu du label. Enfin, quand l’utilisateur rêlache le button gauche du souris, la procédure se termine.

Les trois signaux créent :

self.canvas = ObjectEventService(self.plotting.canvas())  
self.connect(self.canvas, SIGNAL("MouseMove"), self.point\_tracking\_procedure)  
self.connect(self.canvas, SIGNAL("MouseClicked"), self.set\_mouse\_button\_on\_canvas\_clicked)  
self.connect(self.canvas, SIGNAL("MouseReleased"), self.set\_mouse\_button\_on\_canvas\_released)

Voici la procédure de rechercher un point dans le label à modifier quand le button gauche de la souris cliquée:

MouseClicked ?

Self.mouseButtonOnCanvasClicked = True

Self.mouseButtonOnCanvasClicked = True

Lancel une procédure de recherche du point

Si on a trouvé seulement un point

No

Yes

Configuré self.pointToBeModified qui contien des informations pour retrouvé la position du valeur dans la list des valeurs d’ordonnée du label

Une fois que le point dans un label a été bien identifié. Les informations pour retrouver l’indice d’ordonnée dans le label à modifier sont bien mémorisé, dans ce cas, si utilisateur bouge le souris dès ce moment. La procédure ci-dessous va être déclenchée pour réaliser la modification de l’ordonnée du point trouvé:

OperationMode == 2

Yes

Si self.labelToBeModified a été bien configuré, c-a-d que le nom du label et le nom du fichier qui contient telle label.

No

Yes

No

Self.pointToBeModified configuré ?

Yes

Récupérer le ‘plotting context’ dans le modèle selon numéro de page et numéro de table à tracer

Modifié le valeur d’ordonnée à modifier avec les informations dans la variable ‘self.pointToBeModified’

Actualisé l’ensemble des courbes qui représente le label à modifier avec son nouveaux contenu

Enfin, il existe un button afin de laisser utilisateur modifier directement le valeur dans la fichier de la loi anémométrique

## Sauvgardé de la session de travail d’utilisateur

# Conclusion

## Tâches réalisés

/…….

## Developpement en Future

……