

4.3.2.1 *Modèle de données*

Un des principaux problèmes qu'on a rencontré a été la nécessité d'avoir un modèle de données commun à tous les services. Finalement ce modèle a été défini en XML Schéma, un langage permettant de définir des modèles d'objets, et qui nous permet de générer des sources C++ grâce au compilateur XSC.

Ce langage, qui nous permet de créer des objets serialisables en binaire, c'est très important pour récupérer les données en un temps assez court en cas d'échange d'objets et de reprise après une défaillance du système. Le générateur de sources ajoute aussi les méthodes pour récupérer et sauvegarder les classes serialisables dans des fichiers XML.

Ce modèle de données, étant assez grand, a eu plusieurs modifications, avant d'arriver à une situation a peu près stable. On peut remarquer quelques faits importants :

- On n'utilise pas de types basiques du langage, on a créé nos propres types de base, pour éviter des confusions entre unités et des problèmes de précision.
- La compilation des classes n'est pas seulement une transformation des fichiers XSD vers C++, il nous permet aussi de déclarer des fichiers headers «.hpp» pour déclarer des constructeurs par défaut des classes du modèle.
- Les classes dans le modèle de données sont les classes qui vont être utilisées par différents modules. Les classes qui ne sont utilisées que dans un seul module sont appelées données locales, et leur implémentation est effectué par les développeurs du module.

4.3.2.2 *Les modélisations Filtrage et Agenda*

Une des étapes les plus longues dans la conception a été la conception des modélisations Filtrage et Agenda. Le DSE V3R0 définissait la modélisation comme le regroupement d'un profil horizontal et temporel et d'un profil vertical.

Ces deux profils étaient définis comme des volumes continus, ce qui n'est pas envisageable, car le traitement de données complexes est trop complexe. Partant sur cette

base et du code déjà fait dans le prototype et la maquette, on a décidé d'établir des profils discrétisés, c'est-à-dire, comme une liste de points discrets qui modélisent de façon similaire le comportement du profil.

4.3.2.2.1 Différences entre les modélisations Filtrage et Agenda

Les modélisations Filtrage et Agenda sont calculées de façon similaire, mais on peut trouver une première différence : la modélisation Filtrage est plus large que la modélisation Agenda.

La raison de cette différence vient du but du service Filtrage: le but de ce service est de filtrer les vols qui peuvent être présents dans le contrôle d'un vol donné, on laisse hors le filtrage les vols qui ne vont pas être présents. C'est-à-dire que si existe une petite possibilité d'interaction entre deux vols, on doit les inclure dans les filtrages respectifs.

Par contre, le but du service Agenda est plutôt d'afficher des problèmes qui sont susceptibles d'être pris en compte par le contrôleur et de permettre au contrôleur leur manipulation.

On trouve également une extension latérale sur le profil Horizontal et Temporel de la modélisation Filtrage et aussi une extension sur le profil Vertical en haut (montée) ou en bas (descente), qui ne sont pas présentes sur la modélisation Agenda.

4.3.2.2.2 Discrétisation du profil vertical

Ceci est un des points les plus importants de mon travail, car c'est la partie de la modélisation qu'on crée sans partir de la Trajectoire Horizontale OC. Par définition, le profil vertical peut inclure les phases suivants : « Entrée », « Clairée », « Cloche » et « Sortie ».

Chacune de ces phases est un profil vertical, crée sur un faux plan 2D où les abscisses sont ce qu'on appelle les abscisses curvilignes de la vue du vol, et qui représentent la distance parcourue par l'avion depuis l'aérodrome de départ, et les ordonnées sont les différentes altitudes.

Le profil vertical est un enchaînement des différentes phases, qui sont souvent superposées à son plan horizontal. Chacune de ces phases est délimitée par des abscisses et des niveaux de début et de fin de phase, comme montré dans l'exemple de la figure 10:

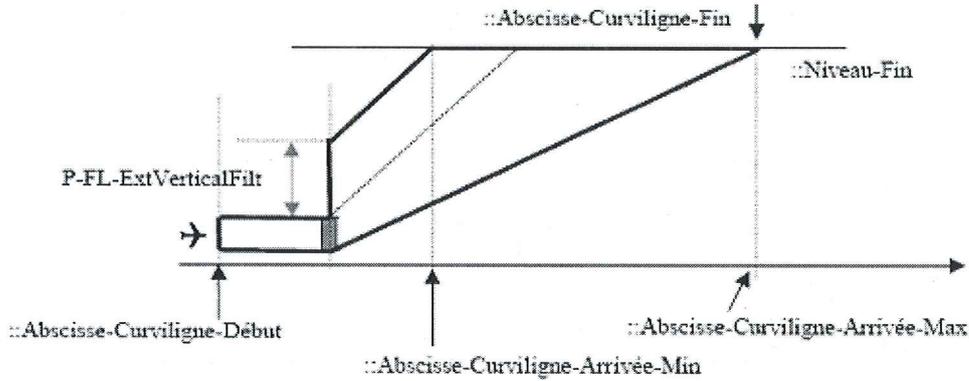


Fig. 10 : Abscisses et niveaux qui délimitent le profil de phase

Etant donné que la détection d'interactions utilise les niveaux occupés par l'avion, et que si un avion est entre deux niveaux, il les occupe tous les deux, on ne s'intéresse pas à un profil de montée (respectivement descente) continu, mais un profil qui montre les niveaux occupés. C'est-à-dire, on cherche les points où la droite de montée (respectivement descente) traverse à un niveau de vol, pour avoir une différence entre les niveaux de vol occupés avant et après ce point. On peut trouver un exemple de ces points dans la figure suivante :

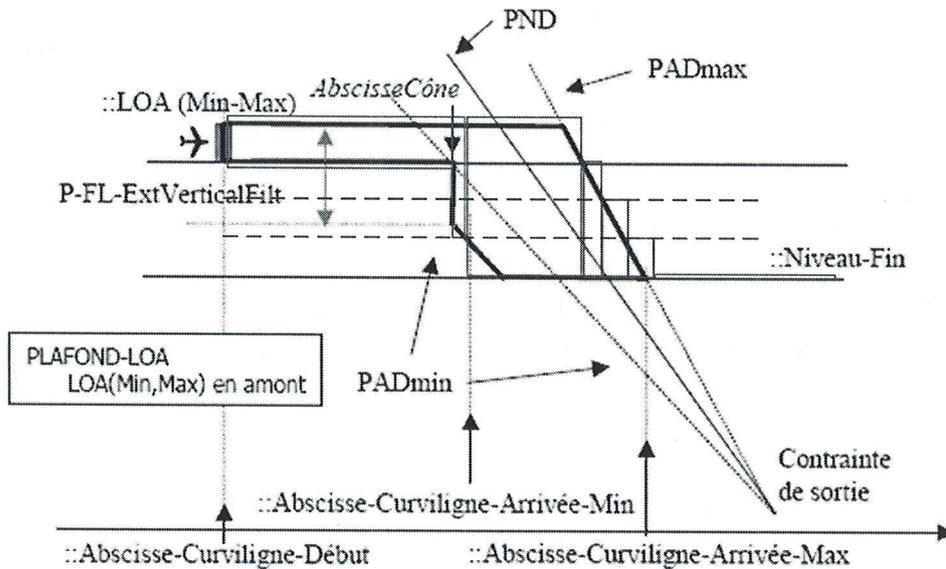


Fig. 11 : Profil discrétisé marqué en rouge

On va sauvegarder ces points d'intersections avec les niveaux dans une structure qu'on appelle abscisse de profil vertical. Cette structure contient l'abscisse curviligne correspondant au point, et les niveaux occupés (minimum et maximum). On considère pour une abscisse curviligne qui est entre deux abscisses de profil vertical, que ses niveaux minimum et maximum sont ceux de l'abscisse du profil vertical qui la précède, comme dans l'exemple de la figure 12.

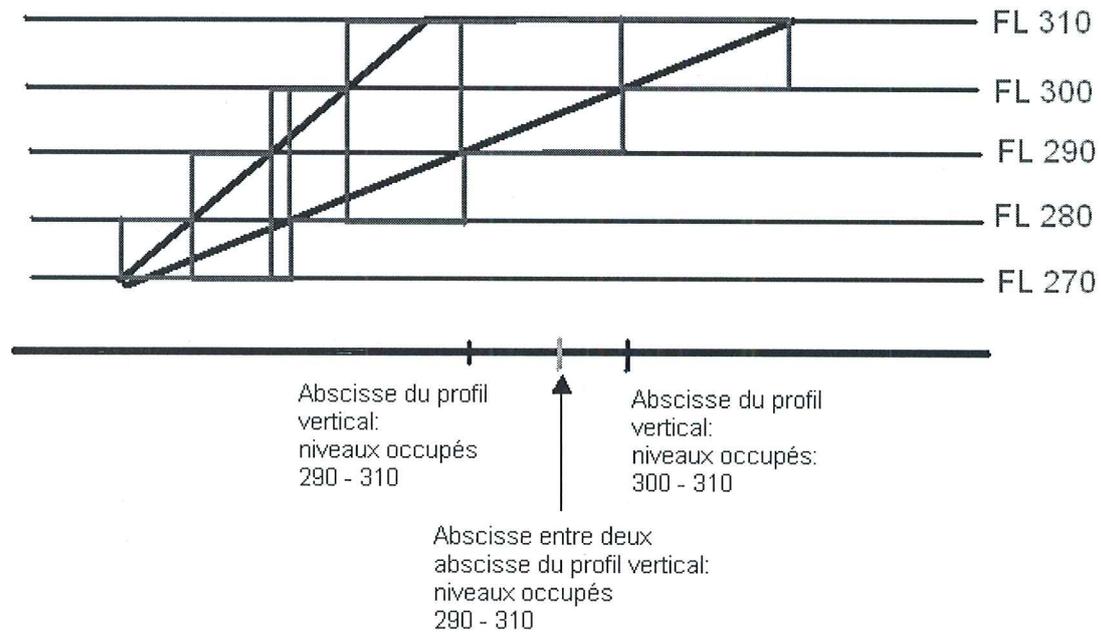


Fig.12 : Exemple de positionnement des altitudes aux abscisses

4.3.2.2.3 Discretisation du profil horizontal et temporel

Dans le cas du profil Horizontal et Temporel, on est parti de l'ensemble des points qui fournissait la Trajectoire Horizontale OC, en les actualisant avec les niveaux minimum et maximum qui leur correspondent, par consultation du profil vertical. On a estimé que ce nombre de points peut ne pas suffire pour une détection d'interactions sûre, et on a décidé d'ajouter des points supplémentaires.

On ajoute :

- Les points définis par les abscisses du profil vertical.
- Des points espacés régulièrement en distance. La distance d'espacement sera fixée après une phase de tests.

4.3.2.2.4 Construction de la modélisation

On a toujours considéré deux possibilités : un objet modélisation qui contient les deux profils dans deux objets différents, ou un objet modélisation qui contient une liste de points avec les informations relatives aux deux profils.

On a finalement décidé d'implémenter la deuxième solution, mais sans abandonner complètement la première, c'est-à-dire, on gardera un objet profil vertical uniquement pendant qu'on calcule la modélisation. Cette deuxième condition était nécessaire, car le calcul du profil vertical se compose des différents calculs des profils verticaux des phases du vol dans la position de contrôle, et certaines de ces phases ont besoin de connaître la phase précédente.

4.3.2.2.5 Construction du Profil Vertical

La construction du Profil Vertical est découpée en quatre fonctions qui construisent les différentes phases du profil et qui sont appelées en fonction des états de ces phases.

Les phases peuvent avoir différents états :

- La phase « Entrée » peut être en état PLANIFIEE, EN_COURS ou TERMINEE.
- La phase « Clairée » peut être en état INEXISTANTE, PLANIFIEE ou EN_COURS.
- La phase « Cloche » peut être en état INEXISTANTE, PLANIFIEE, ABANDONEE ou TERMINEE.
- La phase « Sortie » peut être en état PLANIFIEE ou TERMINEE.

L'ordre d'enchaînement des phases suit le schéma suivant :

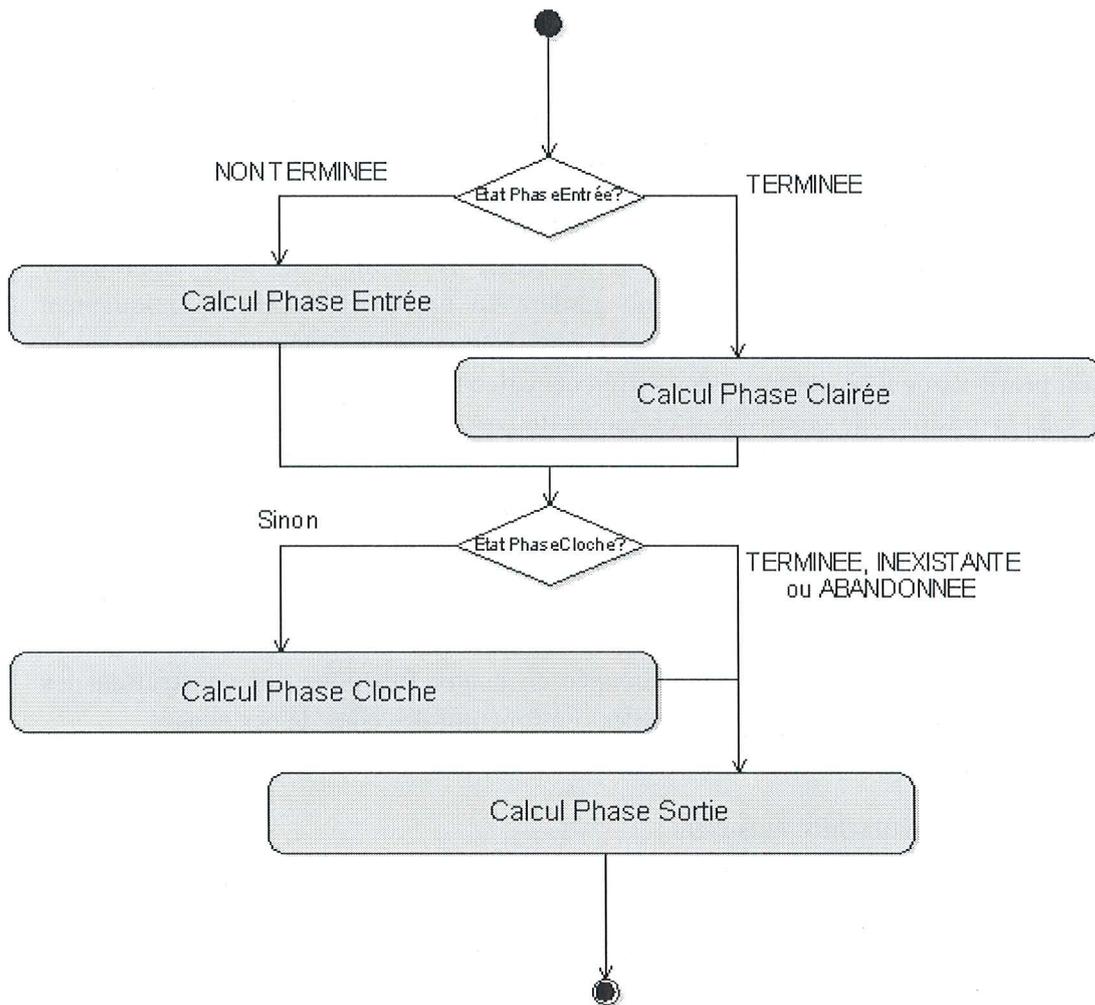


Fig. 13 : Enchaînement des phases du Profil Vertical

On a toujours une phase « Entrée » pour déterminer le profil d'entrée dans la position. Cette phase est toujours calculée avant que le vol soit assumé. Une fois que le vol est assumé, la phase « Entrée » est positionnée à « TERMINEE » et dorénavant le calcul démarre par la phase « Clairée ».

Les phases « Cloche » et « Sortie » ne sont jamais en état « EN_COURS », elles servent uniquement à calculer le possible profil de sortie, mais la vraie sortie se fait par assignation de clairances par le contrôleur, donc par une phase « Clairée ».

4.3.2.1 *Modèle de données*

Un des principaux problèmes qu'on a rencontré a été la nécessité d'avoir un modèle de données commun à tous les services. Finalement ce modèle a été défini en XML Schéma, un langage permettant de définir des modèles d'objets, et qui nous permet de générer des sources C++ grâce au compilateur XSC.

Ce langage, qui nous permet de créer des objets serialisables en binaire, c'est très important pour récupérer les données en un temps assez court en cas d'échange d'objets et de reprise après une défaillance du système. Le générateur de sources ajoute aussi les méthodes pour récupérer et sauvegarder les classes serialisables dans des fichiers XML.

Ce modèle de données, étant assez grand, a eu plusieurs modifications, avant d'arriver à une situation a peu près stable. On peut remarquer quelques faits importants :

- On n'utilise pas de types basiques du langage, on a créé nos propres types de base, pour éviter des confusions entre unités et des problèmes de précision.
- La compilation des classes n'est pas seulement une transformation des fichiers XSD vers C++, il nous permet aussi de déclarer des fichiers headers «.hpp» pour déclarer des constructeurs par défaut des classes du modèle.
- Les classes dans le modèle de données sont les classes qui vont être utilisées par différents modules. Les classes qui ne sont utilisées que dans un seul module sont appelées données locales, et leur implémentation est effectué par les développeurs du module.

4.3.2.2 *Les modélisations Filtrage et Agenda*

Une des étapes les plus longues dans la conception a été la conception des modélisations Filtrage et Agenda. Le DSE V3R0 définissait la modélisation comme le regroupement d'un profil horizontal et temporel et d'un profil vertical.

Ces deux profils étaient définis comme des volumes continus, ce qui n'est pas envisageable, car le traitement de données complexes est trop complexe. Partant sur cette

base et du code déjà fait dans le prototype et la maquette, on a décidé d'établir des profils discrétisés, c'est-à-dire, comme une liste de points discrets qui modélisent de façon similaire le comportement du profil.

4.3.2.2.1 Différences entre les modélisations Filtrage et Agenda

Les modélisations Filtrage et Agenda sont calculées de façon similaire, mais on peut trouver une première différence : la modélisation Filtrage est plus large que la modélisation Agenda.

La raison de cette différence vient du but du service Filtrage: le but de ce service est de filtrer les vols qui peuvent être présents dans le contrôle d'un vol donné, on laisse hors le filtrage les vols qui ne vont pas être présents. C'est-à-dire que si existe une petite possibilité d'interaction entre deux vols, on doit les inclure dans les filtrages respectifs.

Par contre, le but du service Agenda est plutôt d'afficher des problèmes qui sont susceptibles d'être pris en compte par le contrôleur et de permettre au contrôleur leur manipulation.

On trouve également une extension latérale sur le profil Horizontal et Temporel de la modélisation Filtrage et aussi une extension sur le profil Vertical en haut (montée) ou en bas (descente), qui ne sont pas présentes sur la modélisation Agenda.

4.3.2.2.2 Discrétisation du profil vertical

Ceci est un des points les plus importants de mon travail, car c'est la partie de la modélisation qu'on crée sans partir de la Trajectoire Horizontale OC. Par définition, le profil vertical peut inclure les phases suivants : « Entrée », « Clairée », « Cloche » et « Sortie ».

Chacune de ces phases est un profil vertical, crée sur un faux plan 2D où les abscisses sont ce qu'on appelle les abscisses curvilignes de la vue du vol, et qui représentent la distance parcourue par l'avion depuis l'aérodrome de départ, et les ordonnées sont les différentes altitudes.

Le profil vertical est un enchaînement des différentes phases, qui sont souvent superposées à son plan horizontal. Chacune de ces phases est délimitée par des abscisses et des niveaux de début et de fin de phase, comme montré dans l'exemple de la figure 10:

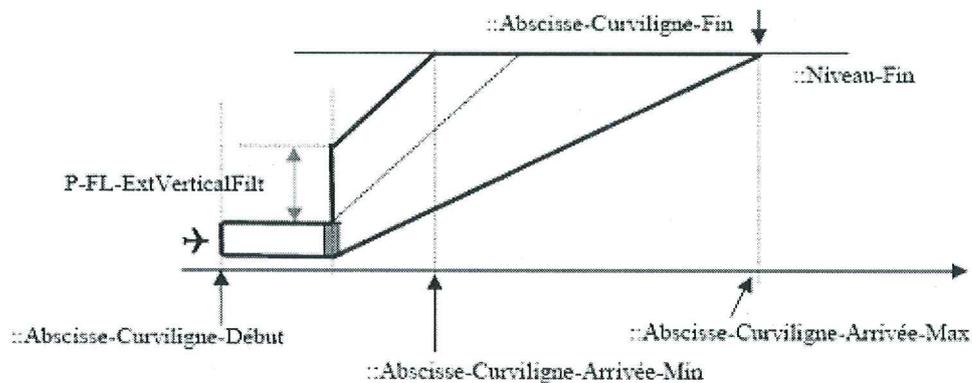


Fig. 10 : Abscisses et niveaux qui délimitent le profil de phase

Etant donné que la détection d'interactions utilise les niveaux occupés par l'avion, et que si un avion est entre deux niveaux, il les occupe tous les deux, on ne s'intéresse pas à un profil de montée (respectivement descente) continu, mais un profil qui montre les niveaux occupés. C'est-à-dire, on cherche les points où la droite de montée (respectivement descente) traverse à un niveau de vol, pour avoir une différence entre les niveaux de vol occupés avant et après ce point. On peut trouver un exemple de ces points dans la figure suivante :

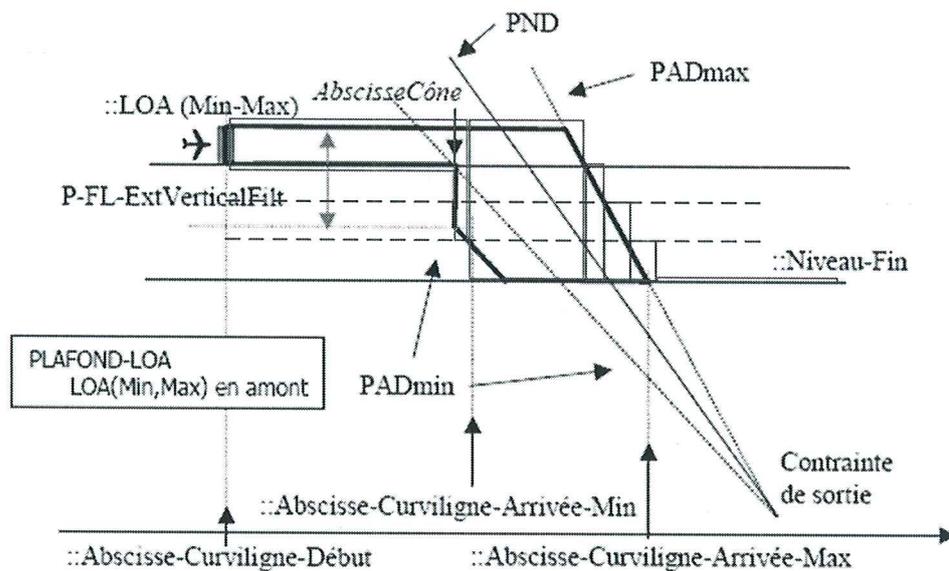


Fig. 11 : Profil discrétisé marqué en rouge

On va sauvegarder ces points d'intersections avec les niveaux dans une structure qu'on appelle abscisse de profil vertical. Cette structure contient l'abscisse curviligne correspondant au point, et les niveaux occupés (minimum et maximum). On considère pour une abscisse curviligne qui est entre deux abscisses de profil vertical, que ses niveaux minimum et maximum sont ceux de l'abscisse du profil vertical qui la précède, comme dans l'exemple de la figure 12.

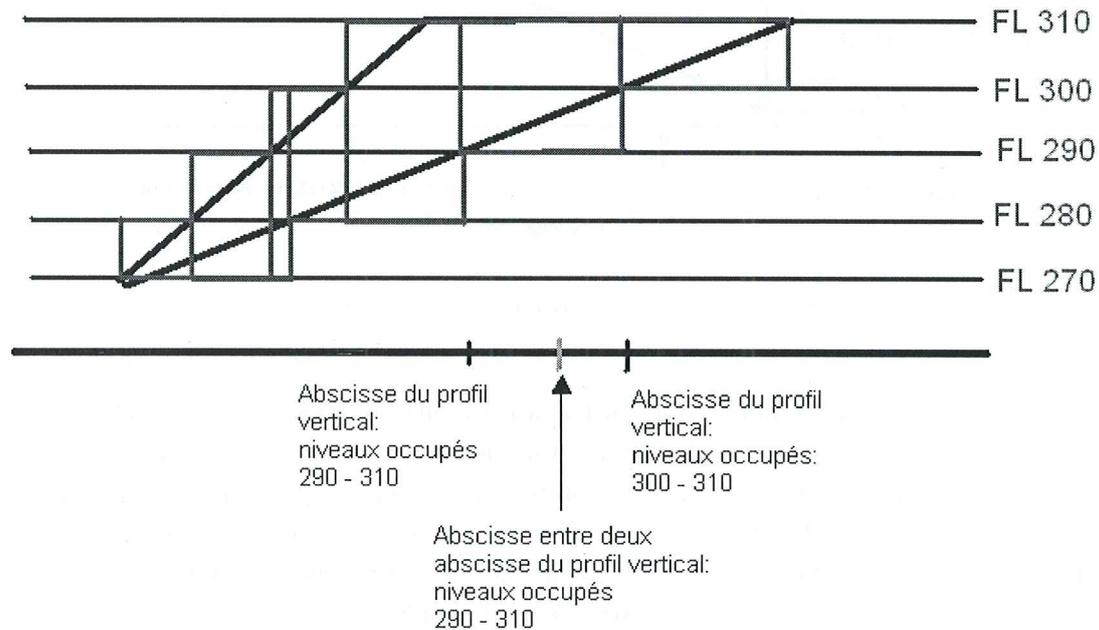


Fig.12 : Exemple de positionnement des altitudes aux abscisses

4.3.2.2.3 Discrétisation du profil horizontal et temporel

Dans le cas du profil Horizontal et Temporel, on est parti de l'ensemble des points qui fournissait la Trajectoire Horizontale OC, en les actualisant avec les niveaux minimum et maximum qui leur correspondent, par consultation du profil vertical. On a estimé que ce nombre de points peut ne pas suffire pour une détection d'interactions sûre, et on a décidé d'ajouter des points supplémentaires.

On ajoute :

- Les points définis par les abscisses du profil vertical.
- Des points espacés régulièrement en distance. La distance d'espacement sera fixée après une phase de tests.