ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAITRISE EN GENIE
M.Ing.

PAR
Gabriel KOUBA

CALCUL DES TRAJECTOIRES UTILISANT LES ALGORITHMES GÉNÉTIQUES EN TROIS DIMENSIONS POUR UN AVION MODÉLISÉ EN SIX DIMENSIONS

MONTRÉAL, LE 22 AVRIL 2010

« Ce n'est pas difficile, d'être une vedette. Ce qui est difficile, c'est d'être un débutant. »

Coluche

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ EVALUÉE

PAR UN JURY COMPOSÉ DE

Mme. Ruxandra Botez, directrice de mémoire
Département de génie de la production automatisée a l'Ecole de technologie supérieure

M. Tony Wong, president du jury
Département de génie de la production automatisée a l'Ecole de technologie supérieure

M. Michel Nadeau Beaulieu, examinateur externe
CAE Inc.

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 28 MAI 2010

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je remercie mon directeur de recherche, le docteur Ruxandra Botez pour m'avoir accueilli dans son laboratoire LARCASE au sein de l'École de technologie supérieure. Durant mes deux ans de maîtrise elle a su se montrer attentive et directive sur mon projet qui a été mené au bout. J'ai pu atteindre un niveau de maîtrise dans le domaine aéronautique grâce à ses conseils et le temps qu'elle m'a consacré.

Je remercie le professeur Sabourin Robert pour m'avoir enseigné les méthodes d'optimisation que j'ai utilisé durant ma maîtrise.

Je remercie le professeur Alain Poirier pour son cours sur la créativité dont j'ai utilisé les principes pour être plus efficace durant ma maîtrise.

Je remercie Mlle Sandrine De-Jésus-Mota, sans qui je ne serai jamais parvenu jusqu'à Montréal pour poursuivre mes études d'ingénieur aéronautique.

Je remercie Fays Julien et Brisemeur Romain, membres et amis du projet dans lequel j'étais assigné et avec qui j'ai travaillé pendant deux ans.

Je remercie mes parents, ma sœur et mon frère pour leur soutien durant toutes mes études.

CALCUL DES TRAJECTOIRES UTILISANT LES ALGORITHMES GÉNÉTIQUES EN TROIS DIMENSIONS POUR UN AVION MODÉLISÉ EN SIX DIMENSIONS

Gabriel KOUBA

RÉSUMÉ

De nos jours en aéronautique, le calcul des trajectoires devient de plus en plus précis. Et pour cause, de lui dépend notre connaissance de la position de l'avion, de sa consommation de carburant, des conflits avec les autres avions, et donc des enveloppes de vol. L'espace aérien étant de plus en plus saturé et les normes concernant la pollution de plus en plus strictes, la précision dans la prédiction des trajectoires est un sujet des plus actuels. Le défaut de la plupart des méthodes utilisées jusqu'à présent est leur limitation au niveau des contraintes qui sont de plus en plus nombreuses. La problématique du projet consiste dans le calcul d'une trajectoire aéronautique en trois dimensions (altitude, longitude, latitude) pour un avion modélisé en six dimensions (déplacements spatiaux en altitude, longitude, latitude et angles de roulis, tangage, lacet) en fonction de multiples contraintes (viabilité de l'appareil, ajustement d'altitude…). Afin de répondre à ce problème nous avons utilisé la méthode d'optimisation par algorithmes génétiques. L'avantage de tels algorithmes est que l'on peut optimiser un problème suivant un grand nombre de contraintes pouvant provenir de disciplines différentes (dans notre cas : structure de l'avion, corrections de trajectoires...). Les résultats obtenus ont permis de tracer des trajectoires optimisées suivant six contraintes. 1) le rapprochement de points de rendez-vous, 2) l'ajustement d'altitude, 3) l'évolution de la route horizontale, 4) l'angle d'arrivée sur les points de rendez-vous, 5) la viabilité de l'appareil, et 6) le non-dépassement de trajectoire…

Mots-clés: Algorithmes génétiques, trajectoire, flight management system, contrainte de vol.

TRAJECTORIES CALCULATIONS WITH GENETC ALGORITHMS ON THREE DIMENSIONS FOR AN AIRCRAFT IN SIX DIMENSIONS

Gabriel KOUBA

ABSTRACT

Nowadays in aeronautics, the calculation of trajectories becomes more and more precise. In fact, with this higher precision, comes our knowledge of aircraft's position, its fuel's consumption, and conflicts with other planes: in other words its flight's envelope. The flight space becomes more and more saturated and the standards concerning the pollution tighter, the precision in trajectories' prediction is a burning issue. The disadvantage of most methods used until now is their limitation concerning the number of the constraints, which are more and more numerous. The purpose of the project consists in computing of an aeronautical trajectory in three dimensions (latitude, longitude, altitude) for an aircraft modelized in six dimensions (spatial movements in latitude, longitude, altitude and angles of roll, pitch, yaw) according to multiple constraints (viability of the device, the adjustment of altitude). In order to answer this problem we used a method of optimization by genetic algorithms. The advantage of such algorithms is that we can optimize a problem following a large number of constraints which can result from different disciplines (in our case: structure of the aircraft, corrections of trajectories). The obtained results allowed us to draw optimized trajectories following six constraints. 1) the link of waypoints, 2) the adjustment of altitude, 3) the evolution of the horizontal road, 4) the angle of arrival on the waypoints, 5) the viability of the aircraft, 6) the non-overtaking of trajectory …

Keywords: Genetics algorithms, trajectory, flight management system, flight constraint.

TABLE DES MATIERE

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

INTRODUCTION

Le projet concerne les techniques de la navigation aérienne. Celle-ci a su évoluer au fil des années en fonction des besoins et des technologies disponibles. Tout commença à la naissance de l'aéronautique où il fallut très rapidement savoir s'orienter et se diriger lors d'un vol. L'expérience du calcul de trajectoire reposait alors sur la navigation maritime qui fut donc naturellement le parent de la navigation aérienne. Cependant, la navigation aérienne a évoluée et évolue encore plus rapidement à cause de plusieurs facteurs. Le premier facteur est la vitesse de l'avion, qui est souvent supérieure à la vitesse des bateaux. Le deuxième facteur est l'ajout d'une dimension supplémentaire au déplacement aérien qui est l'altitude. Les calculs des trajectoires devaient se faire plus rapidement et précisément que ceux des trajectoires maritimes. En effet, l'autonomie de l'aviation dépend toujours de la consommation de carburant. C'est ainsi que l'on a vu l'introduction du navigateur, communément appelé « le troisième homme ». Le travail de cette personne était de fournir au pilote le plan de vol et de calculer la route, la consommation, l'horaire, etc. L'arrivée de l'électronique et de l'informatique supprima le navigateur dans la plupart des avions. On assista donc à l'arrivée de FMS (Flight Management System) ou de l'instrumentation avionique, ayant comme entrées les différents points de passages obligatoires de l'avion, et comme sorties, les trajectoires de vol et la consommation du combustible.

Ces calculateurs sont les vrais héritiers des navigateurs et cette évolution a été possible grâce aux données disponibles et surtout aux avancées technologiques de ce siècle. Au début du $20^{ième}$ siècle le navigateur se basait sur des données de vol restreintes, généralement la vue, l'origine du vol ou encore la radio. Ses calculs utilisaient des compas, règles, montres, rapporteurs, etc.… Les limites d'un tel modèle étaient donc humaines. Mais l'arrivée des calculateurs sur le marché a permis la conception de méthodes mathématiques plus fiables et complexes, s'effectuant à la vitesse des processeurs. De la même manière les données de vol provenaient de sources plus nombreuses, tel que la radio, les instruments de bord, le personnel au sol et aussi les satellites (figure 1.1). Ce type de données est plus difficile à

gérer par un seul homme, que par des calculateurs. Les limites des calculs des trajectoires n'ont pas été de nature humaine ou électronique mais surtout informatique.

Au fil des années, les algorithmes ont évolué. On ne demandait plus uniquement au calculateur de trouver une trajectoire particulière, on attendait que cette trajectoire soit optimisée. Il fallait que les méthodes soient capable de prendre en considération des paramètres comme la consommation du combustible, la structure de l'avion ou encore la stabilité de la trajectoire. Il n'est pas évident d'augmenter le nombre de contraintes. Très vite l'on arrive à des systèmes sur-contraints qui ne sont pas capables de fonctionner parfaitement et assez rapidement. C'est pour cette raison qu'aujourd'hui le calcul de trajectoires reste des plus actuels et nécessaire. On a besoin d'obtenir des trajectoires plus précises.

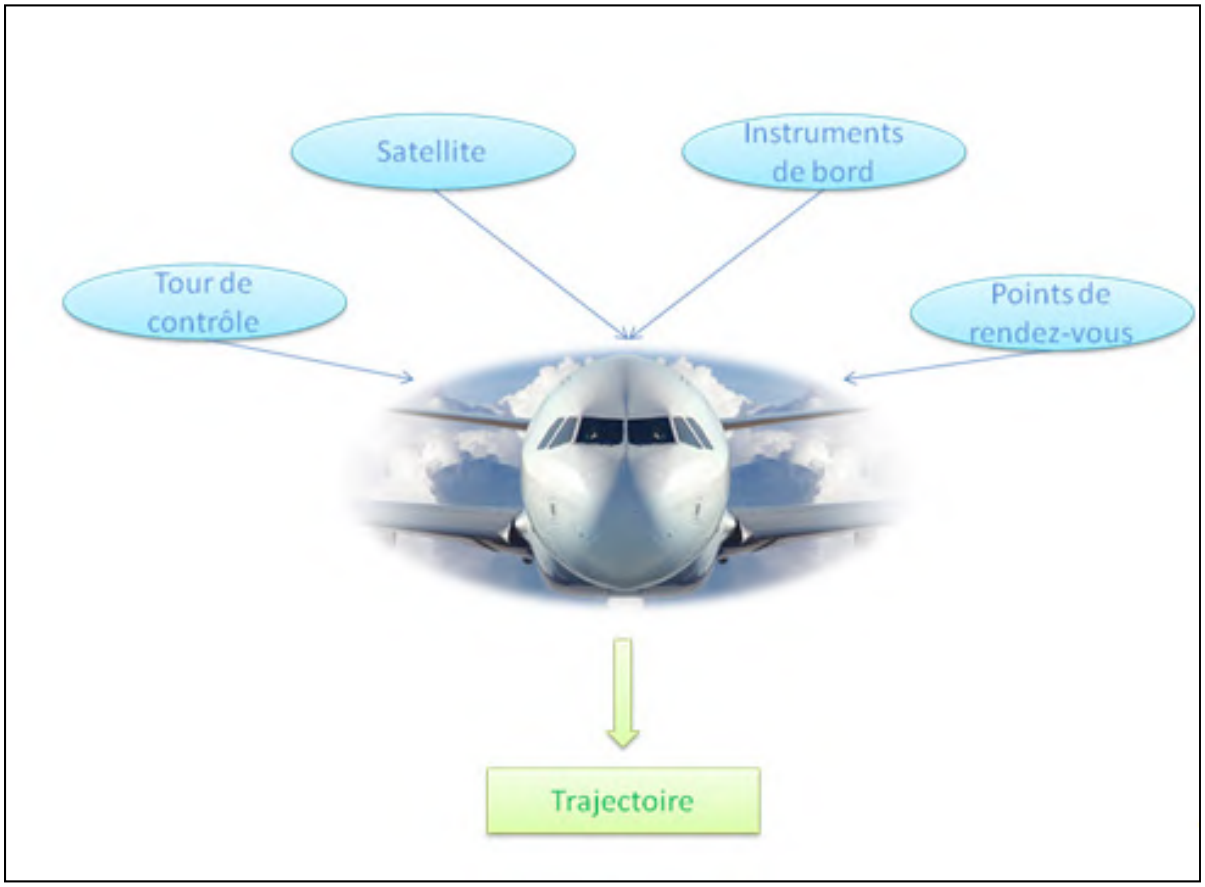

Figure 1.1 Données disponibles pour l'avion.

CHAPITRE 1

PROJET ET CONTEXTE

1.1 Problématique et hypothèses

La problématique du projet est de calculer une trajectoire en trois dimensions (latitude, longitude et altitude) pour un avion modélisé en six dimensions (spatiales : altitude, longitude, latitude et angulaires : roulis, tangage, lacet). Voici les hypothèses du projet : (Leurs détails et leurs justifications seront explicités une fois celle-ci énoncées)

- Le modèle doit passer par une série de points de rendez-vous en 3D donnés par l'utilisateur tel qu'il est actuellement fait dans les avions.
- Le modèle doit calculer le passage par les points de rendez-vous en fonction des points de rendez-vous précédents et suivants afin de diminuer la longueur du trajet total.
- Le modèle de l'avion est un point qui peut avoir des angles de lacet, roulis et tangage maximum de 30 degrés (les raisons de ce choix sont explicitées prochainement dans ce chapitre).
- Le modèle doit se stabiliser le plus rapidement possible en altitude.
- Le gradient de latitude doit être égal au gradient de longitude (cette définition permet d'établir une route horizontale droite entre les points de rendez-vous).
- Les évolutions du modèle en tangage doivent être stables.
- Le modèle doit éviter les dépassements des trajectoires qui rallongeraient la route horizontale, par exemple, dépasser la latitude demandée par le point de rendez-vous suivant, pour revenir dessus ensuite.

Ces hypothèses ont été considérées pour plusieurs raisons. Il a fallu tout d'abord délimiter le projet afin de ne pas s'écarter et de partir dans trop de directions. Nous avons donc établi dans un premier temps un modèle aéronautique assez simple. Bien que le projet suggère l'avion dans sa structure et ses comportements aérodynamiques, il aurait été trop ambitieux de démarrer sur une méthode non maitrisée avec un modèle trop complexe.

L'angle maximum de 30 degrés a été choisi de manière fixe, mais cependant il dépend de l'avion, de la vitesse, de l'altitude. En effet il devrait varier selon le modèle aéronautique que l'on décide d'appliquer. Pour notre étude il a été choisi de fixer cette valeur à 30 degrés afin de se concentrer sur l'implémentation de la méthode et non sur le modèle aéronautique. Il sera possible de jouer sur cette variable quand la méthode sera maitrisée.

La variation de l'altitude est déjà établie dans les trajectoires des avions civiles, où le changement de l'altitude demande un changement de poussée qui peut influencer négativement le confort des passagers, ou la surconsommation de carburant. Il faut atteindre la nouvelle altitude le plus rapidement possible. De même les dépassements de trajectoires augmentent la longueur du trajet à parcourir, la consommation du combustible et la durée du vol.

Le système de points de rendez-vous est actuellement celui en vigueur dans les FMS (Flight Management System). Il s'agit d'entrer la série de points de rendez-vous configurant la trajectoire désirée par le pilote. Notre programme fonctionne sans normaliser les aéroports et les points de rendez-vous en vigueurs.

1.2 Revue de littérature

Nous présentons ici la revue de littérature des différentes méthodes de calcul des trajectoires. Premièrement nous présentons les méthodes de calculs utilisant les algorithmes génétiques. Ces méthodes ont déjà été largement appliquées dans le monde de l'aéronautique. Par exemple dans les conflits aériens [2], cet algorithme solutionne des problèmes de très grandes envergures. En effet, ils ont les capacités de résoudre des systèmes de plus de 1,500 avions

en très peu de temps. Cette capacité est donc très appropriée pour couvrir un grand espace de solutions. Et pour cause, en 1960, John Holland a étudié les systèmes évolutifs, en se basant sur la théorie de la sélection naturelle de Darwin, afin de mettre au point les algorithmes génétiques [4]. Les algorithmes génétiques résultent donc directement de l'évolution des espèces, ce qui montre leur capacité à s'adapter aux problèmes. C'est donc pour ces arguments que nous avons choisi cette méthode, contrairement à d'autres méthodes de contrôle. Cette méthode a déjà été utilisée dans le calcul de trajectoire pour des cas bien précis [1, 10]. Dans ces différentes études [1, 10], il y a l'exemple d'un missile suivant la trajectoire d'un avion. Il s'agit uniquement d'une trajectoire qui vise à se rapprocher de la trajectoire d'un avion, ce qui est semblable à se rapprocher d'un point de rendez-vous. Les plus grandes différences notées, consiste dans le fait que les points de rendez-vous ne bougent pas (pour un missile qui suit un avion, le point à suivre évolue).

D'autres méthodes furent étudiées comme celles utilisant la logique floue [12] qui permettent aussi de contrôler la trajectoire. En effet, au lieu de chercher des points permettant de construire une trajectoire, on contrôle directement son évolution en fonction des paramètres (rapprochement de points de rendez-vous, consommation en combustible) établis par l'utilisateur [8]. Le problème qui fut discriminatoire envers cette méthode aurait été les rajouts de trajectoires [5], ce qui n'était pas une option pour des trajectoires civiles. Un autre facteur discriminatoire fut le fait que la construction du modèle demandait une simulation de trajectoire qu'il fallait ensuite contrôler, ce qui demandait plus de ressources. C'est pour ces raisons que les réseaux de neurones n'ont pas été utilisés [6], donnant des résultats similaires à la logique floue.

La question des dépassements de trajectoires et du nombre d'hypothèses éliminèrent la méthode de la descente [3].

Une autre méthode qui fut écartée, est celle d'interpolation linéaire entre les points de rendez-vous. Cette méthode obligeait d'augmenter beaucoup le nombre de points de rendez-vous afin d'éviter les dépassements. De plus, les polynômes générés avaient des degrés très élevés, les rendant par la même occasion très instables.

La deuxième partie de notre revue de littérature concerne les calculateurs de trajectoires actuels et ce qu'ils offrent. Ces calculateurs sont les FMSs (Flight Management System's) [7], et ils affichent les trajectoires en deux dimensions (la route horizontale et l'altitude sont sur des écrans séparés). La nouvelle génération commence à voir apparaître des affichages sur le FMS en trois dimensions. De plus, les effets des nombreux paramètres considérés semblent difficiles à gérer car les trajectoires sont souvent sur-contraintes. L'introduction de nouvelles méthodes de calculs pour construire les trajectoires de vol fut donc notre préoccupation. Les informations dont dispose un FMS pour fonctionner sont les points de rendez-vous en quatre dimensions (longitude, latitude, altitude et vitesse de passage), et la consommation voulue du combustible. Il fallait donc utiliser un algorithme ayant la possibilité de rajouter des variables ou contraintes aux trajectoires comme taux de descente ou de montée. Cette possibilité est présente dans la méthode des multiplicateurs de Lagrange [5], pourtant son utilisation en deux dimensions uniquement fut discriminatoire.

Plusieurs études ont été réalisées pour modifier les algorithmes génétiques, et calculer les meilleures conditions initiales par contrainte de l'espace des solutions [11]. Il s'agit d'éviter la dispersion des recherches de la solution désirée. Ceci permet la convergence plus rapide de l'algorithme, en risquant la perte d'une partie des solutions. Cependant, pour réaliser une telle méthode il faut dans un premier temps disposer d'un algorithme fonctionnel. Bien que la rapidité de convergence soit très importante à bord de l'avion, ce problème n'était cependant pas une priorité dans notre simulation. Pour ces différentes raisons, cet aspect fut laissé de côté jusqu'au démarrage du projet futur.

Plusieurs travaux ont enfin été réalisés à l'aide des algorithmes génétiques (AG) pour générer des Approches par Descentes Continues (ADC) [9]. La particularité de ces descentes sans paliers pour l'avion est le nombre élevé de paramètres qu'il faudrait gérer, et donc de contraintes à satisfaire.

La plupart des méthodes déjà implémentées dans les FMS ne sont pas capables de générer de telles trajectoires. On présente ici une méthode qui a été validée et qu'il faut généraliser ou étendre pour les modèles de la trajectoire générale de l'avion.

1.3 Mémorisation des trajectoires

Les seules données dont le programme ait besoin pour fonctionner est une série de points de rendez-vous, ce qui implique plusieurs hypothèses et solutions. En effet, les points doivent être viables et correspondre aux points de rendez-vous d'un avion en vol.

L'angle de montée demandé par l'utilisateur lors de la prise en compte des points de rendez-vous est primordial. Si cet angle dépasse 30 degrés, le programme n'est pas conçu pour tracer une trajectoire car elle ne répondrait plus aux hypothèses de notre problématique. Il existe des moyens pour palier à de tels problèmes, comme par exemple l'ajout de points de rendez-vous supplémentaires et ainsi la création de trajectoires circulaires ou en spirale [5] pour éviter de dépasser l'angle de 30 degrés, ce qui est montré dans la figure 1.2. Mais ceci implique de ne plus respecter l'hypothèse de route droite entre les points de rendez-vous puisqu'au lieu de droites il y aura des spirales. Comme le montre la figure 1.2, on ne peut pas joindre de façon viable les points 2 et 3. On rajoute donc une courbe (partie entourée) qui ne lie pas directement ces points 2 et 3, ce qui est du à l'acceptation de points *non viables* (dont l'angle de montée dépasse 30 degrés) et ne fait pas partie de ce projet.

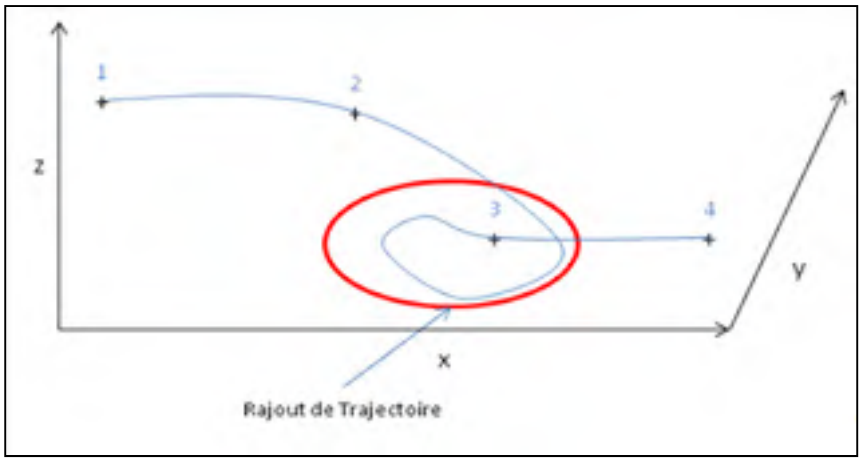

Figure 1.2 Exemple de rajout de trajectoire.

L'angle de route (uniquement dans le plan horizontal) n'est pas soumis à ce calcul. En effet, si l'avion ne peut pas effectuer de virages à plus de 30 degrés, rien ne dit que le plan de vol n'admette pas des différences supérieures à 30 degrés pour la route. Ceci obligera l'avion à

effectuer plusieurs manœuvres de réajustement à la suite, chacune ne dépassant pas 30 degrés.

1.4 Points de rendez-vous fictifs et réels

Afin de répondre aux besoins d'une trajectoire intelligente, les points de rendez-vous sont recalculés après que l'utilisateur les ait fournis à l'algorithme. Il s'agit uniquement de les ajuster pour que la trajectoire générale demandée soit plus courte. Il s'agit ici de simuler le fait que l'avion n'est pas obligé de passer par les points de rendez-vous afin de raccourcir le trajet total [7].Chaque point fictif est calculé en fonction de son prédécesseur et de son suivant. Pour ceci nous avons introduit un taux de proximité entre le point recalculé (point fictif) et le vrai point. Ce taux, exprimé en pourcentage, équivaut au rapprochement que l'on veut entre le vrai point de rendez-vous donné par le pilote et celui calculé (point fictif). Par défaut nous sous-entendons par un taux de 100%, que ces deux points seront les mêmes. Le taux est défini comme sur la figure 1.3.

Le nouveau point de rendez-vous (points fictif) 2' se trouve à l'intersection de deux droites (figure 1.3). La première est parallèle à l'axe des ordonnées *y* et passe par le point de rendez-vous 2 considéré. La deuxième passe par les deux points formés par le taux (celui de la figure 1.3 est de 85% en exemple) de proximité et les droites entre les points de rendez-vous, comme le montre la figure 1.3.

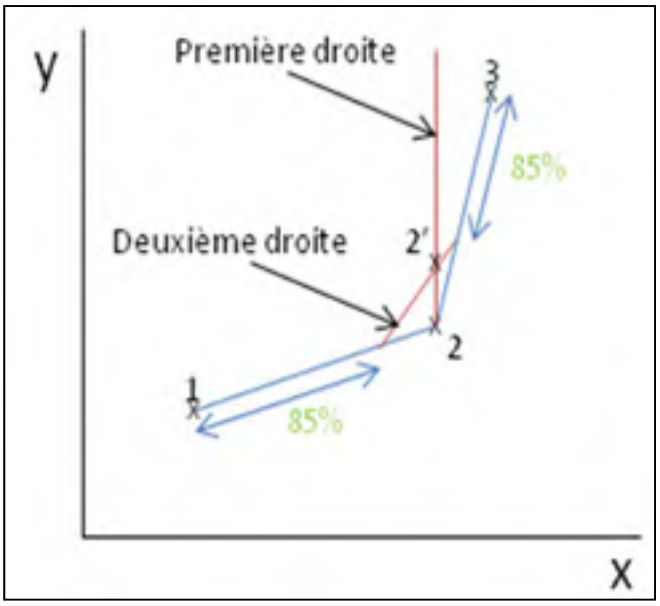

Figure 1.3 Calcul du point de rendez-vous fictif 2'
à partir du point de rendez-vous réel 2.

Voici un exemple montré dans la figure 1.4, de comparaison entre la route d'origine (en traits pleins), et la route recalculée (en pointillé) pour un taux de 85%. Cet exemple démontre que ces routes ne sont pas tout à fait les mêmes et que les différences pourraient conduire à des problèmes. Cependant avec un taux correct (supérieur à 80%), les deux trajectoires se collent rapidement.

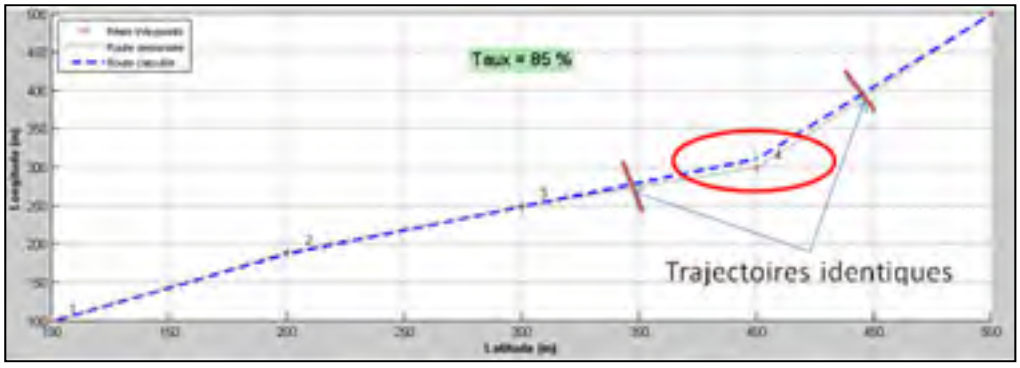

Figure 1.4 Différence de route causée par la différence
entre les points de rendez-vous fictifs et réels.

1.5 Modélisation de l'avion

Comme nous l'avons énoncé dans les hypothèses, la modélisation de l'avion est un point caractérisé par six paramètres : latitude, longitude altitude, roulis, tangage, lacet, et ne nécessite pas des équations complexes. Le choix de cette vision fut poussé par l'avancement initial du projet. En effet l'accent a été mis sur la maitrise des algorithmes génétiques dans un premier temps. Cependant se limiter aux trois dimensions spatiales n'aurait pas permis de pousser le projet aussi loin; en variant sur un plus grand nombre de paramètres, nous obtenons des trajectoires plus réelles et contrôlées. Sans cet ajout, il aurait été difficile de montrer la dominante aéronautique du projet.

1.5.1 Composantes spatiales

Les composantes spatiales regroupent la latitude, la longitude et l'altitude. Contrairement à un problème en trois dimensions, il s'agit de mettre l'accent sur la partie aéronautique. Pour ce faire, il y a deux composantes à considérer : la route de l'avion dans le plan horizontal et l'altitude.

Premièrement nous considérons la route de l'avion et donc le plan horizontal. Celle-ci représente le plan horizontal et bien que les algorithmes génétiques permettent de varier les trois dimensions en même temps, il n'est pas nécessaire de les lier entre elles. En d'autres termes, même si les résultats, et les calculs se font en trois dimensions, il n'y a pas de liens entre la route horizontale et l'altitude. La deuxième composante est donc l'altitude. Celle-ci doit être considérée séparément du déplacement horizontal car elle n'a pas les mêmes influences sur le système. En effet le principal moyen d'influencer la variation d'altitude d'un avion est la variation de la poussée, ce qui n'est pas le cas de la route (la variation horizontale de l'avion) qui nécessite des surfaces de contrôle (empennages verticaux ou horizontaux). La poussée, et implicitement la consommation de carburant, est un problème à part et c'est pourquoi l'évolution de l'altitude se calcule séparément de la route horizontale. De même, le trafic aérien se gère essentiellement par des différences d'altitude pour éviter la

collision de deux appareils. L'altitude est donc une problématique, la route horizontale en est une autre.

1.5.2 Composantes angulaires

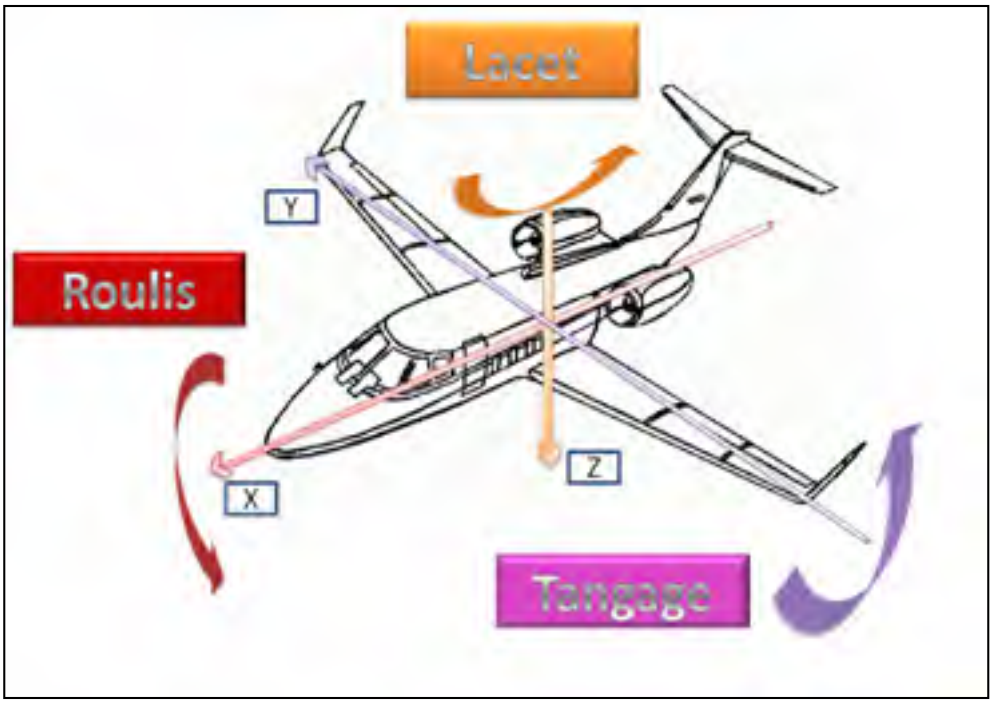

Figure 1.5 Angles de l'avion sur les trois axes *x* (roulis), *y* (tangage) et *z* (lacet).

Les composantes angulaires de la vitesse sont le lacet (z), le tangage (y) et le roulis (x) montrées sur la figure 1.5. La différence avec les composantes spatiales vues précédemment est que les composantes angulaires sont interdépendantes. Le roulis dépend du tangage et du lacet et le tangage dépend du lacet. Ainsi la variation entre ces trois paramètres est très importante car la moindre variation de l'un peut faire fluctuer tout le système. Le calcul de chacune de ses composantes s'effectue entre deux points de trajectoires à l'aide des matrices de rotations (1.1), (1.2) et (1.3).

Soit ψ l'angle de rotation (lacet) autour de l'axe *z*, on obtient la matrice de rotation autour de ψ,

$$R_z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.1}$$

Soit θ l'angle de rotation (tangage) autour de l'axe *y*, on obtient la matrice de rotation autour de θ,

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{1.2}$$

Soit ϕ l'angle de rotation (roulis) autour de l'axe *x*, on obtient la matrice de rotation autour de ϕ,

$$R_x(\varphi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix} \tag{1.3}$$

Pour obtenir la matrice de rotation totale de l'avion et ainsi calculer les composantes angulaires, il faut multiplier ces trois matrices, et obtenir ainsi l'équation (1.4), comme suit :

$$R(\psi,\beta,\gamma) = R_z(\psi)R_y(\theta)R_x(\varphi) =$$

$$\begin{pmatrix} \cos\psi\ cos\ \theta & \cos\psi\ sin\ \theta\ sin\ \varphi - \sin\psi\cos\varphi & \cos\psi\ sin\ \theta\cos\varphi + \sin\psi\sin\varphi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\varphi + \cos\psi\cos\varphi & \sin\psi\sin\theta\cos\varphi - \cos\psi\sin\varphi \\ -sin\ \theta & cos\ \theta\ \sin\varphi & \cos\theta\cos\varphi \end{pmatrix} \tag{1.4}$$

Un programme a été implémenté pour obtenir ces trois composantes en tout point de la trajectoire et ainsi décrire le mouvement de notre avion. On utilise la distance entre deux points de trajectoire et cette matrice de rotation pour retrouver les trois angles nécessaire à la description du modèle.

CHAPITRE 2

LA DESCRIPTION ET L'IMPLÉMENTATION DES ALGORITHMES GÉNÉTIQUES

2.1 Principe et exemple

Les Algorithmes Génétiques AG sont utilisés comme des méthodes d'optimisation permettant de déterminer l'optimum d'une fonction dans un espace donné. Dans notre cas, l'espace est l'environnement de l'avion dans ses trois dimensions, celui-ci étant lui-même décrit en six dimensions. La fonction rassemble toute les trajectoires possibles pour l'avion et son optimum est la trajectoire optimale que l'on désire.

Les AG sont basées sur le principe de la sélection naturelle, et ont le but de trouver une solution optimale à un problème. Il s'agit d'une méthode itérative. Le principe fondamental peut se résumer de cette manière: on prend aléatoirement une population initiale, équivalente à une série de solutions possibles pour notre problème. On teste ensuite ces solutions et on détermine si elles répondent au problème. Plus une solution répond correctement au problème, plus elle a de chance d'être utilisée dans l'itération suivante. Pour parvenir à l'itération suivante, on va créer une nouvelle population (population enfants) à partir de la population précédente (population parents). Pour créer cette nouvelle population on croise les meilleures solutions de la population précédente et on fait muter certaines solutions. On recommence ensuite le processus d'évaluation de la nouvelle population. Ce principe est résumé dans la figure 2.1. Ceci est l'algorithme de base.

L'opération de croisement peut s'expliquer en se basant sur les fondements de la sélection naturelle. Pour illustrer cette opération, concrètement on prend l'exemple d'un homme ayant les yeux verts et les cheveux bruns. Celui-ci pourrait provenir d'une mère aux cheveux bruns et d'un père aux yeux verts. Lors de la reproduction (le croisement) des parents, la solution des yeux verts et des cheveux bruns s'est transmise à l'individu fils. Plusieurs mécanismes de reproduction et de sélection des gènes se sont mis en évidence dans cette opération. Les

cheveux bruns et les yeux verts sont peut-être plus adaptés dans l'espace de vie du fils. Cette solution optimale existait déjà mais était dispersée chez les parents. L'avancement d'une génération à permis l'obtention de cet optimum.

De même la mutation au niveau des AG s'explique par la mutation génétique au niveau des individus biologiques.

Bien que le principe puisse paraître trivial dans une société ou la sélection naturelle est bien établie dans l'évolution des espèces, celui-ci n'a commencé ses débuts en optimisation mathématique qu'en 1960, avec les travaux de John Holland de l'Université du Michigan [4]. L'idée de transposer les méthodes de résolutions construites par la nature en méthodes d'optimisation mathématique est assez récente. Il ne s'agit pourtant là encore que d'un fragment de ce que la nature fait en réalité.

Les AG de bases passent donc par plusieurs étapes fondamentales pour obtenir la solution optimale d'un problème dans un espace donné. Ces étapes sont les suivantes :

- sélection de la population parente de départ;
- évaluation de la population parente;
- croisement et mutation des individus de la population parente pour créer la population enfant;
- évaluation de la population enfants;
- si la population enfant correspond à la solution recherchée, on arrête l'algorithme;
- si la population enfant n'est pas celle recherchée, la population enfant devient la population parente et on recommence.

Plusieurs méthodes pour l'amélioration de cet algorithme sont possibles à chaque étape dépendamment de l'appréciation de l'utilisateur et du problème posé. Nous considérons par exemple l'élitisme, qui permet de garder les meilleurs membres d'une génération à une autre. Ces méthodes et les choix qu'elles impliquent seront détaillés dans les sections suivantes.

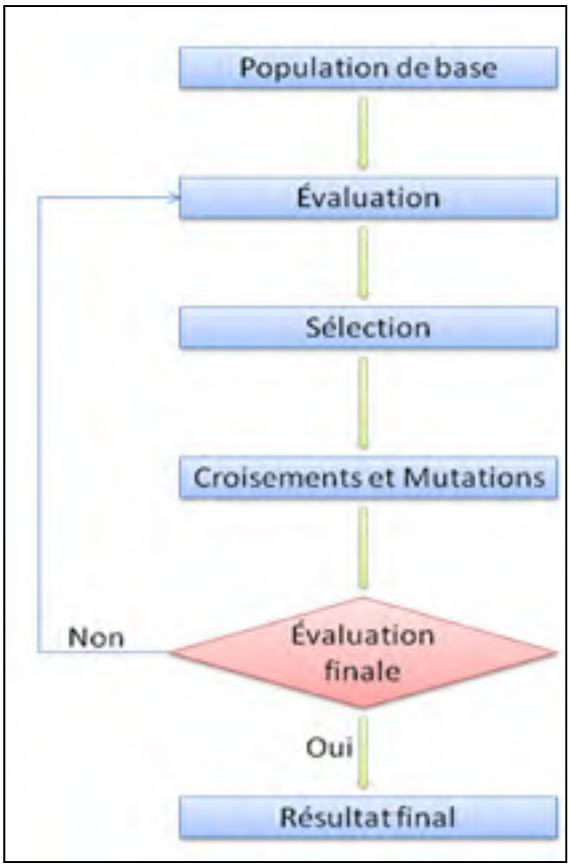

Figure 2.1 Récapitulatif des algorithmes génétiques.

Nous allons rapidement analyser un exemple mathématique afin d'illustrer la méthode. Avant de se lancer dans le calcul de trajectoire, nous avons considéré un exemple simple afin de maitriser la méthode. Il s'agissait de trouver le maximum d'une fonction en 3 dimensions : $z=x+y$ ou x et y sont définis de 0 à 128 en binaire, x étant les abscisses, y les ordonnées et z les côtes. Cet exemple est en effet donné en binaire et utilise des chiffres de 7 bits (ceci est un choix arbitraire pris pour l'exemple). y et x pouvant valoir au maximum 128 (nombre maximum possible pour des valeurs en 7 bits), le but est donc de trouver un z valant 256. Il s'agissait ici de mettre au point la méthode et son évolution, c'est pourquoi on connait déjà la solution. L'exemple a volontairement été pris simple, pour étudier l'évolution des algorithmes génétiques.

Initialisation et première évaluation

La population de base comprend 6 chromosomes ou individus de deux fois 7 bits pris aléatoirement comme le montre le tableau 2.1. Il s'agit ici de prendre 12 nombres en binaire totalement au hasard. Leur évaluation est ensuite effectuée en additionnant *x* et *y* pour obtenir *z* et en comparant les *z* entre eux afin de trouver celui qui répond le mieux au problème, qui est ici de trouver le *z* le plus grand. Chaque *z* est ensuite divisé par la somme des *z* et multiplié par 100 pour les passer en pourcentage. Par exemple pour la première ligne, on a comme valeur *x*=0110110 ce qui vaut 54 en décimal. On obtient la une valeur de *y* en décimal de 102. On obtient donc une valeur pour *z* de 102+54 qui vaut 156. De la même façon on calcul la valeur de *z* pour les 6 individus et en les additionnant tous on obtient 1185. Une fois tous les *z* calculées, on les passe en pourcentage. 156 équivaut donc ici à 13 pourcent (car 154*100/1185=13).

De plus, l'individu 1 a pour valeur 0110110 dans le tableau 2.1. Ceci est bien un nombre binaire pris entre 0000000 et 1111111. Ces individus sont pris sur 7 bits car ils ont un maximum décimal de 128 (= $2^0+2^1+2^2+2^3+2^4+2^5+2^6$).

Tableau 2.1 Population initiale de six individus

Individus parents	x							y							Évaluation des z (%)
1	0	1	1	0	1	1	0	1	1	0	0	1	1	0	13
2	1	0	1	0	1	1	0	1	1	0	1	1	1	1	23
3	0	1	0	0	1	0	1	1	1	1	1	0	1	0	16
4	0	1	1	0	0	1	0	0	1	1	1	1	0	1	17
5	1	0	1	0	0	1	1	0	1	1	0	0	1	0	18
6	0	0	1	0	0	1	1	1	0	0	0	0	0	0	13

Croisement

Le but ici n'est pas de détailler toutes les méthodes de croisement et les choix qu'elles impliquent. Ceci sera fait dans la partie de croisement du projet. On effectue pour cette exemple un simple croisement pour les *x* et un autre pour les *y* (sans mélanger les *x* et les *y*). La probabilité d'un chromosome d'être choisi pour le croisement est directement proportionnel à son évaluation. Ainsi les meilleurs chromosomes se reproduisent pour donner la nouvelle population enfant du tableau 2.2 dans lequel nous avons surligné les endroits de croisement aléatoires deux à deux. Pour croiser deux nombres parents, on les coupe à des endroits aléatoires et on les recolle en entremêlant les bouts coupés. Ces bouts sont appelé *gènes*. Il est indiqué après chaque enfant entre parenthèse dans le tableau 2.2, les parents dont il est issu. Surligné est représenté une partie d'un parent et laissée en blanc la partie du deuxième parent.

Par exemple dans le tableau 2.2, l'individu enfant de la ligne 1 descend des individus parents du tableau 2.1 à la ligne 3 et 4. Le croisement s'est effectué après le deuxième bit comme montré par le sur-lignage du tableau 2.2. Ainsi l'individu 1 du tableau 2.2 est le collage des deux premiers bits de l'individu 3 et des 5 derniers bits de l'individu 4 du tableau 2.1.

Tableau 2.2 Population après les croisements

Individus enfants (parents)	x							y						
1 (3 et 4)	0	1	1	0	0	1	0	1	1	1	1	1	0	1
2 (3 et 4)	0	1	0	0	1	0	1	0	1	1	1	0	1	0
3 (2 et 4)	1	1	1	0	0	1	0	1	1	1	1	1	0	1
4 (2 et 4)	0	0	1	0	1	1	0	0	1	0	1	1	1	1
5 (3 et 2)	0	0	1	0	1	1	0	1	1	0	1	1	1	1
6 (3 et 2)	1	1	0	0	1	0	1	1	1	1	1	0	1	0

Mutation

On applique ensuite une mutation aléatoire sur chaque bit d'une probabilité de 1/100. Cette valeur dépendant du nombre d'individus et de la sensibilité du système. Le résultat est montré sur le tableau 2.3 et les mutations sont surlignées. Ici la mutation consiste à remplacer un 0 par un 1 ou un 1 par un 0. On effectue ensuite une évaluation de la population enfant. Dans le tableau 2.3 la mutation a déjà été effectuée. On reprend donc le tableau 2.2 de croisement, en faisant muter un bit du paramètre y de l'enfant 6. Celui-ci valait 1, il vaut maintenant 0.

Tableau 2.3 Population après les mutations

Individus enfants	x							y							évaluation des z (%)
1	0	1	1	0	0	1	0	1	1	1	1	1	0	1	19
2	0	1	0	0	1	0	1	0	1	1	1	0	1	0	14
3	1	1	1	0	0	1	0	1	1	1	1	1	0	1	15
4	0	0	1	0	1	1	0	0	1	0	1	1	1	1	19
5	0	0	1	0	1	1	0	1	1	0	1	1	1	1	19
6	1	1	0	0	1	0	1	1	1	1	1	0	0	0	14

Itérations et critère d'arrêt

Toutes ces opérations (croisement, mutation et évaluation) constituent une itération. Une fois la première itération effectuée on la réitère jusqu'à ce que l'algorithme stagne, ou que l'on ait atteint un certain nombre d'itérations (donné par l'utilisateur). Le fait de se croiser et de muter va faire avancer la population vers la solution recherchée. Dans cet exemple, le nombre de chromosomes était faible. Il a donc fallu beaucoup d'itérations pour résoudre le problème donné et maximiser la fonction. Mais au bout de 144 itérations, l'algorithme nous donne le résultat de 255, ce qui est presque 256 (le maximum recherché). Sur la figure 2.2 nous montrons tout l'espace de solutions (la

plaque représente *z*) et l'évolution des solutions calculées (points noirs) par l'algorithme génétique jusqu'à la solution finale (le *z* le plus grand). La fonction à bien été maximisée.

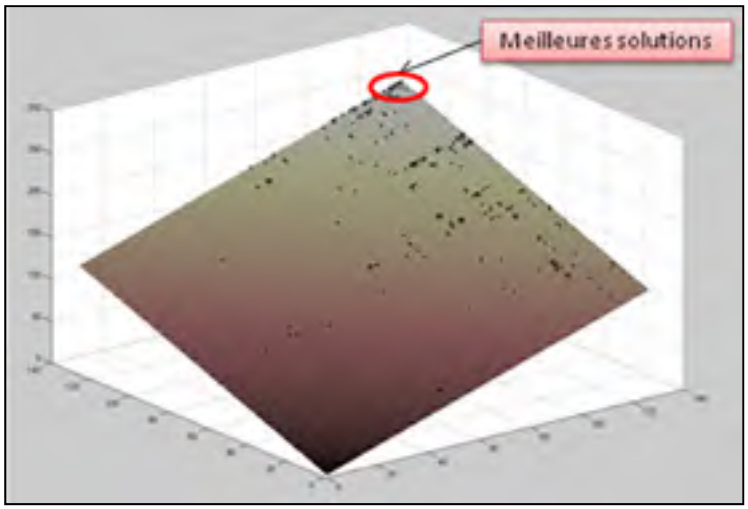

Figure 2.2 Représentation graphique de l'évolution de l'exemple *« z=x+y »*.

Cet exemple implémenté et compris fut un tremplin pour le reste du projet.

2.2 Sélection de la population initiale

Le choix de la population initiale est primordial, car celle-ci contient les germes de notre solution finale. Il est donc de notre devoir d'implanter l'algorithme dans les bonnes conditions, pour que celui-ci trouve rapidement la bonne solution. L'efficacité de l'algorithme dépend en partie de la population initiale. Si l'on ne sait rien sur la position de la solution, il est naturel de générer aléatoirement la population initiale. Le but est de s'assurer que tout l'espace soit bien couvert pour ne pas mettre de côté un espace où la solution pourrait se trouver, ce qui ralentirait énormément l'algorithme.

Il faut de plus s'assurer que les individus de la population initiale respectent les contraintes de la solution optimale. Ces contraintes sont contenues dans la phase d'évaluation, mais pour une plus grande efficacité, elles doivent être prises en compte dès le début. Les contraintes du

projet réunissent les angles permis à l'avion, l'ajustement de l'altitude, et bien d'autres, et seront détaillées dans la section 2.3 de l'évaluation.

Dans notre cas, on connait le point de départ et le point d'arrivé de l'avion. Imaginons qu'il se déplace entre deux points de rendez-vous. Le point de départ est le point *A* (latitude 100 m, longitude 100 m, altitude 100 m) et le point d'arrivé est le point *B* (latitude 200 m, longitude 190 m, altitude 120 m). On obtient donc la solution finale de l'algorithme génétique (le point *B*) si on part du point *A*, le principe du calcul de trajectoire étant de contrôler l'évolution de l'algorithme pour obtenir cette solution.

L'algorithme va donc générer aléatoirement une population initiale, autour du point de départ *A*, pour démarrer la première itération. Les résultats obtenus par notre algorithme pour la population initiale, générée autour du point *A*, sont donnés dans le tableau 2.4. Dans une population normale, pour faire fonctionner notre algorithme, il y a 400 individus, mais pour des raisons de visualisation, on ne montre que les résultats obtenus pour 10 individus dans les tableaux de ce chapitre.

Tableau 2.4 Population initiale de dix individus

Individus	Latitude (m)	Longitude (m)	Altitude (m)
1	109	105	104
2	103	102	106
3	109	105	109
4	105	105	103
5	102	100	105
6	101	107	104
7	105	102	108
8	101	103	107
9	105	105	103
10	102	107	108

Si l'on prend l'exemple des latitudes, celles-ci doivent être comprises entre 100 et 200 mètres (entre le point *A* et le point *B*). Ne voulant pas aller directement sur la solution afin de permettre une évolution contrôlée de la trajectoire, les individus de la population initiale doivent rester dans une échelle acceptable. Dans notre cas, nous avons choisi la différence entre le point *B* et le point *A*, divisée par 10. Pour l'exemple en question la latitude on se retrouve alors entre 100 et 110 mètres. On fait de même pour les longitudes.

Dans le cas des altitudes c'est différent. On sait que l'altitude doit progresser plus rapidement que la longitude et la latitude. On peut donc augmenter la fenêtre d'initialisation. On a donc ici la différence entre les altitudes des points *B* et *A*, divisée par 2.

2.3 L'évaluation

L'évolution des *AG* passe par la fonction d'évaluation. C'est elle qui saura valider ou non chaque individu et lui permettra de se reproduire. Lorsqu'une population satisfait la fonction d'évaluation, chacun de ses individus est alors noté en fonction de sa capacité à être une bonne solution, c'est-à-dire qu'il faut compléter chaque contrainte de la fonction d'évaluation. Cette note déterminera son avenir dans la population. C'est en somme une épreuve de passage. Un individu qui a une mauvaise note, aura une probabilité forte d'être éliminé par le processus de sélection. Il reste cependant intéressant de le conserver même si celui-ci n'est que peu admissible, car un de ses gènes, une partie de lui, peut générer un individu enfant de bonne qualité.

Dans notre cas, notre fonction d'évaluation comprend six sous-fonctions d'évaluation différentes décrivant l'avion sous différents paramètres. Encore une fois, il ne s'agit pas ici uniquement d'arriver au point de rendez-vous suivant mais aussi de contrôler l'évolution de la trajectoire, et, alors d'obtenir une trajectoire optimale.

Une fois les six sous-fonctions d'évaluation calculées (distance euclidienne, ajustement de l'altitude, évolution de la latitude et longitude, dépassement, viabilité, angle horizontal d'arrivée), il faut les rassembler en une fonction d'évaluation, définissant chaque individu

pour la génération suivante de l'algorithme. Cette fonction est une note ou un marquage pour chaque individu, afin de savoir s'il est une solution se rapprochant de la solution optimale ou non. Les sous-sections suivantes détaillent une après l'autre les six sous-fonctions d'évaluation.

2.3.1 La distance Euclidienne

Cette sous-fonction d'évaluation permet de quantifier la distance entre deux points en trois dimensions [10]. Le but est donc de noter la distance d'un individu par rapport au point d'arrivée, mais aussi de s'assurer qu'il s'en rapproche pour l'atteindre et ne part pas dans une direction opposée (sa distance Euclidienne serait alors négative). Nous cherchons donc à minimiser cette sous-fonction d'évaluation. Dans le cas d'une distance Euclidienne négative, l'individu est éliminé.

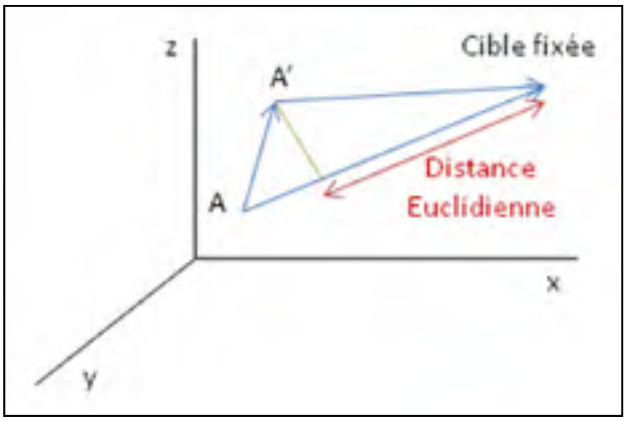

Figure 2.3 Fonction d'évaluation de la distance Euclidienne.

Il est intéressant d'analyser l'évolution d'une trajectoire soumise à cette sous-fonction d'évaluation (figure 2.3 où *A* est le point de départ et *A'* est le point évalué). Ce test permet de juger l'importance de la sous-fonction d'évaluation des la distance euclidienne et de

remarquer ses effets secondaires. Ceci est réalisé dans l'optique d'obtenir un meilleur contrôle sur la trajectoire.

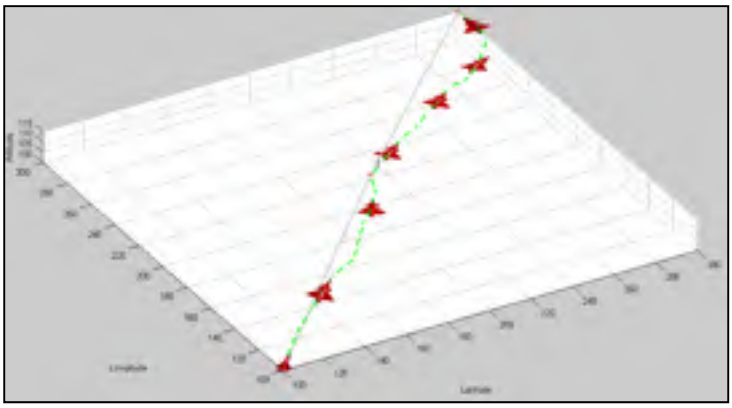

Figure 2.4 Trajectoire soumise à la fonction d'évaluation de la distance Euclidienne.

On peut remarquer (figures 2.4 et 2.5) que le but de l'algorithme ici est d'atteindre le point de rendez-vous suivant. Il ne tient pas en compte de la viabilité de la route de l'avion. Il rejoint donc les points de rendez-vous de façon chaotique.

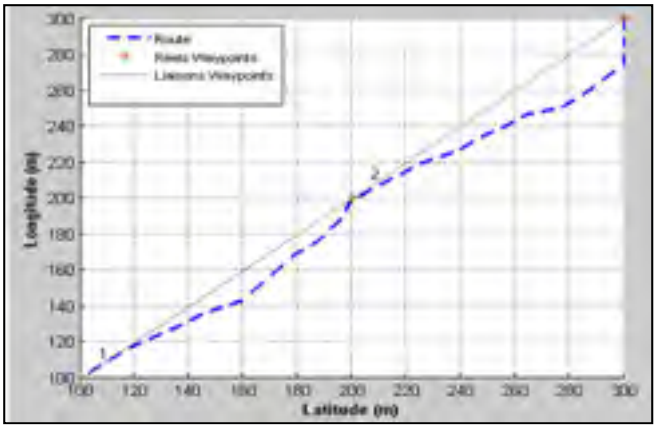

Figure 2.5 Route calculée avec la sous-fonction d'évaluation de la distance Euclidienne.

2.3.2 L'ajustement de l'altitude

L'altitude devant être ajustée avant la latitude et la longitude, il faut donc avoir une sous-fonction d'évaluation (figure 2.6) qui ordonne les points en fonction de leur évolution en altitude. Plus ils évoluent rapidement vers la hauteur demandée, plus la sous-fonction d'évaluation sera meilleure. Nous cherchons donc à minimiser cette sous-fonction d'évaluation.

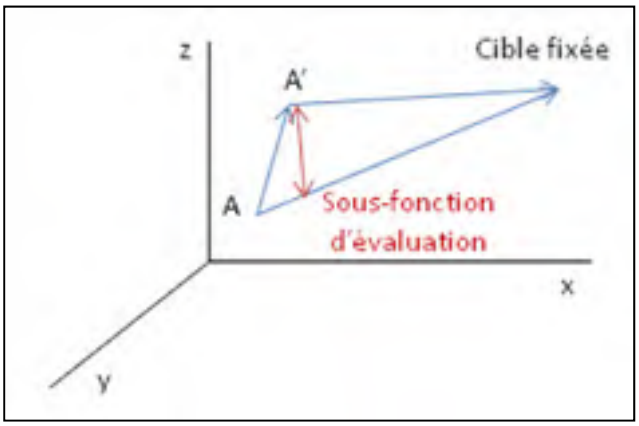

Figure 2.6 Fonction d'évaluation de l'ajustement d'altitude.

Nous avons également calculé une trajectoire soumise à cette sous-fonction d'évaluation seulement (figures 2.7 et 2.8). Cependant, seule l'altitude est contrôlée et donc les évolutions sur le plan horizontal restent aléatoires (figure 2.8). Comme on peut le remarquer, l'algorithme évolue trop vite en latitude (axe des *x*). Par contre il évolue pour atteindre le plus tôt possible le palier en altitude. Seule cette sous-fonction joue sur l'évolution de l'altitude, elle est donc primordiale. Les discontinuités présentes en altitude sont dues au non-contrôle du tangage, le gradient de l'altitude n'est donc pas constant. Dans les figures 2.7 et 2.8, les points de rendez-vous fictifs et réels sont les mêmes car il ne s'agit pas ici de tester l'option qui recalcule les points de rendez-vous. De plus, la courbe en traits continus représente la courbe calculée par nos algorithmes génétiques.

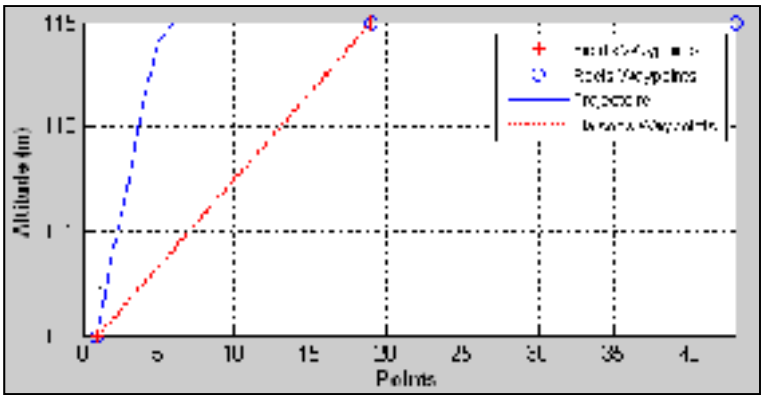

Figure 2.7 Altitude versus le point de trajectoire calculée avec la fonction d'évaluation de l'ajustement d'altitude.

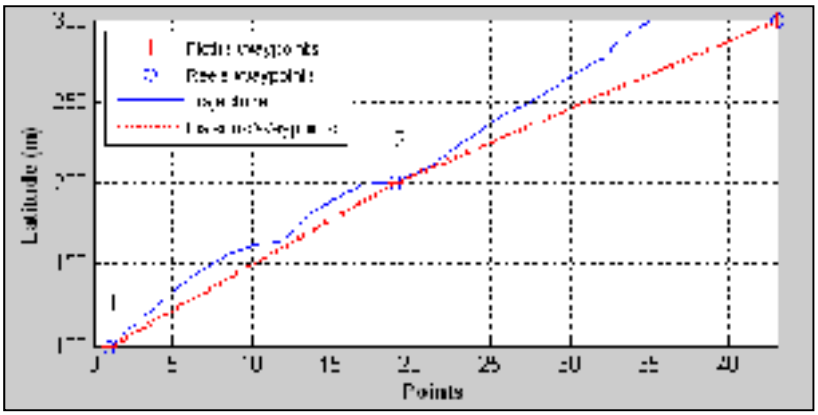

Figure 2.8 Latitude versus le point de trajectoire calculée avec la fonction d'évaluation de l'ajustement d'altitude.

2.3.3 Évolution de la latitude et la longitude

La latitude et la longitude possèdent le même gradient d'évolution pour éviter les instabilités de trajectoires sur le plan horizontal ce qui donne l'équation (5). Ainsi horizontalement la trajectoire suivra la droite tracée par les deux points de rendez-vous qu'elle lie.

Soit, X la latitude, Y la longitude, A le premier point de rendez-vous, B le second point de rendez-vous, A' le point évalué, Δx la variation de latitude entre les deux points de rendez-

vous *A* et *B*, *Δy* la variation de longitude entre les deux points de rendez-vous *A* et *B*, *δx* la variation de latitude entre le premier point de rendez-vous *A* et l'individu évalué *A'*, *δy* la variation de longitude entre le premier point *A* de rendez-vous et l'individu évalué *A'*, on a, l'équation (2.5) et la figure 2.9.

$$\delta x/\Delta x = \delta y/\Delta y \qquad (2.5)$$

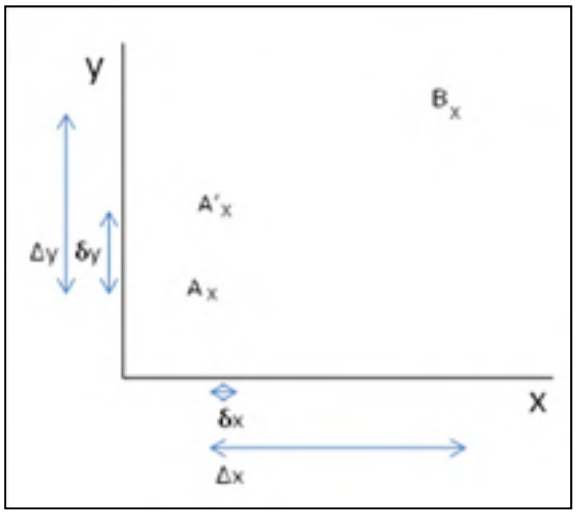

Figure 2.9 Fonction d'évaluation de la latitude versus la longitude.

Le rapport des deux parties de l'équation (2.5) doit donc se rapprocher de 1.

En appliquant uniquement cette sous-fonction d'évaluation, on obtient les figures 2.10, 2.11, 2.12 et 2.13. Bien que l'on puisse remarquer une certaine instabilité sur l'évolution de la trajectoire, on peut aussi voir que la route calculée (en trait plein) correspond à la route désirée. Ceci est du au fait que l'algorithme ne s'occupe que d'une chose ici : coller la route horizontale (en pointillé).

L'évolution de l'altitude est catastrophique car on ne s'en occupe pas. La viabilité angulaire n'est même pas présente. Cette sous-fonction est particulièrement importante car lorsque l'on dit de faire évoluer la latitude et la longitude on demande également une progression de

trajectoire pour atteindre le point de rendez-vous. Elle remplie donc indirectement une partie du rôle de la sous-fonction d'évaluation liée à la distance Euclidienne.

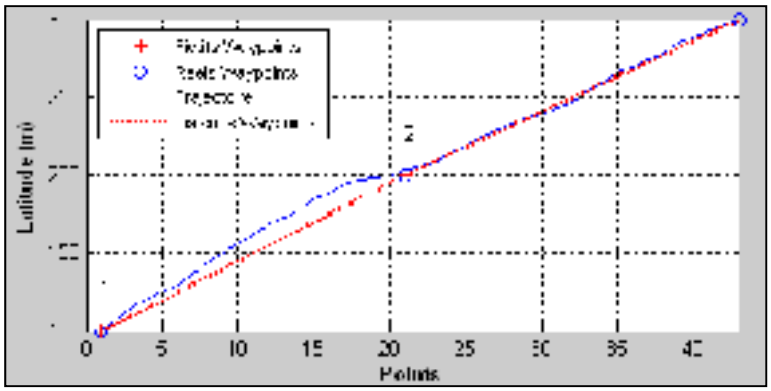

Figure 2.10 Latitude versus le point de trajectoire calculée avec la fonction d'évaluation de latitude/longitude.

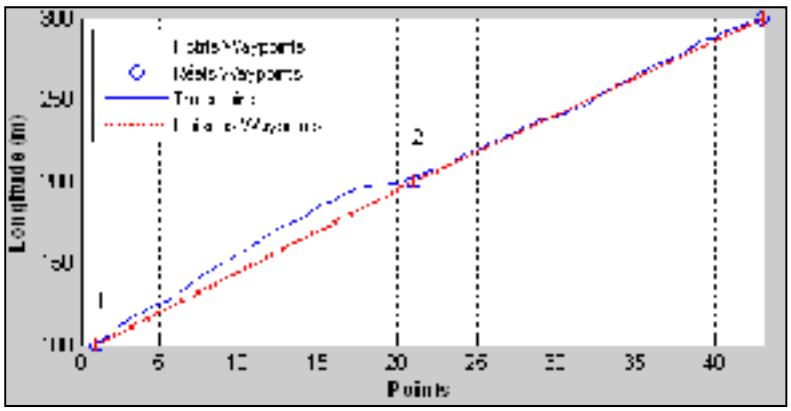

Figure 2.11 Longitude versus le point de trajectoire calculée avec la fonction d'évaluation de latitude/longitude.

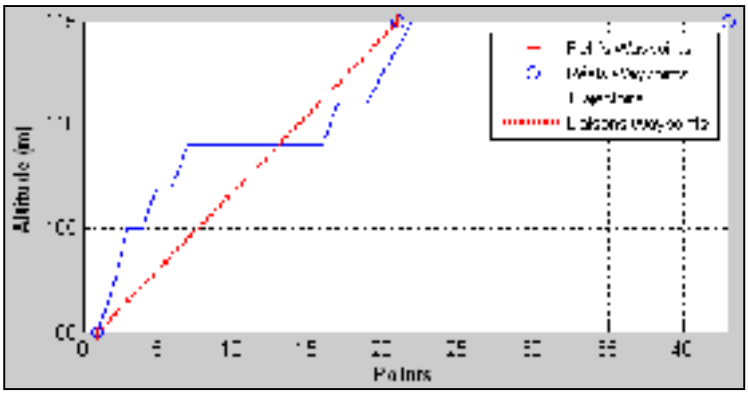

Figure 2.12 Altitude versus le point de trajectoire calculée avec la fonction d'évaluation de latitude/longitude.

Dans la figure 2.13 on ne voit pas la route demandé en gris car elle est confondue dans la route calculée en tiret.

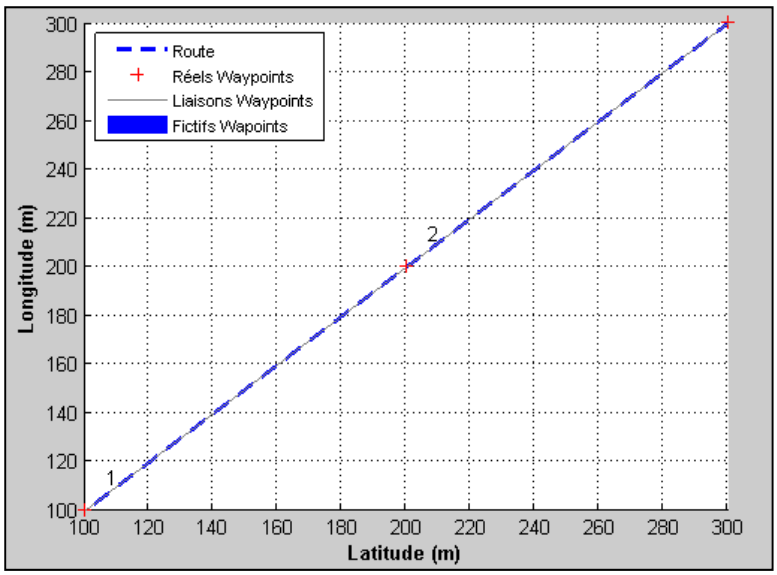

Figure 2.13 Route soumise à la fonction d'évaluation de latitude/longitude.

2.3.4 Le dépassement

Il s'agit ici de la première sous-fonction d'évaluation binaire. Une sous-fonction d'évaluation binaire est une sous-fonction d'évaluation éliminatoire. Elle n'a que deux valeurs, 1 ou 0. Si l'individu rempli les conditions elle se met à 1 et il survivra. Sinon, il est directement effacé sans conditions (si elle est donc à 0). Les enjeux sont trop importants et pourraient mettre en péril la cohérence même de la trajectoire. Ici nous parlons de dépassement de trajectoire dans le plan horizontal. L'altitude étant gérée à part car elle évolue plus vite, nous parlons uniquement du plan horizontal.

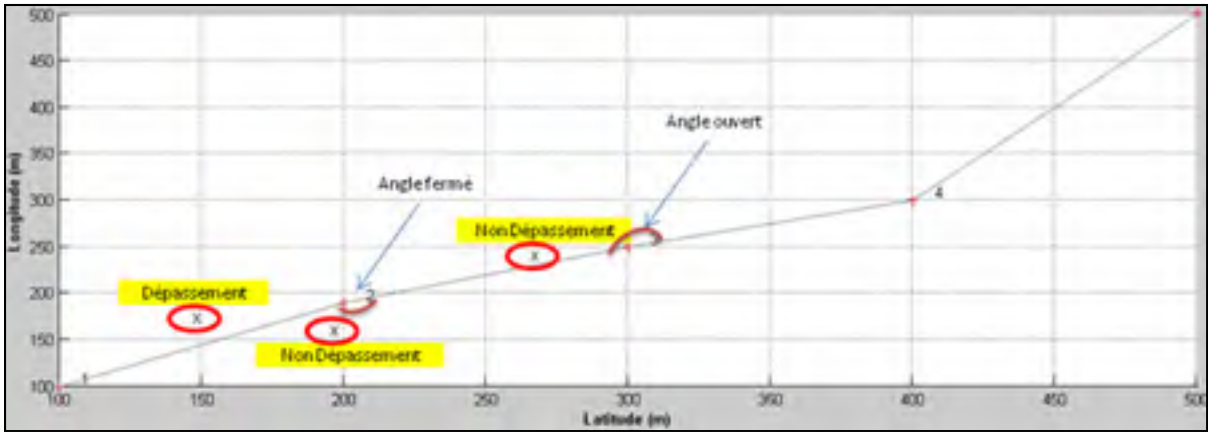

Figure 2.14 Fonction d'évaluation de dépassement.

Plaçons-nous au niveau de l'angle formé par trois points de rendez-vous (par exemple l'angle 2 sur la figure 2.14). Le dépassement est autorisé uniquement si cet angle est ouvert ce qui signifie, que cet angle est supérieur à 180 degrés. Si celui-ci est fermé (inférieur à 180 degrés), le dépassement est interdit. Ceci est résumé sur la figure 2.14.

Le problème que peut poser le raisonnement d'une sous-fonction d'évaluation éliminatoire concerne l'extinction de toute une population. En effet, sachant que nous éliminons des individus, il est possible que si la population n'est pas élevée, elle soit complètement éliminée. Il y a deux moyens de palier à cette éventualité. Le premier moyen est d'augmenter le nombre d'individus pour qu'il en survive assez. Le deuxième moyen est de compter sur

l'incidence indirecte d'autres sous-fonctions d'évaluation pour éviter les dépassements. La sous-fonction de dépassement n'est donc plus la seule à opérer la contrainte de dépassement.

Ce rôle est rempli par les sous-fonctions d'évaluation sur l'évolution de la latitude et la longitude et sur la distance Euclidienne. Si les points sont pris en fonction de ces deux sous-fonctions d'évaluation, ils auront tendance à se coller sur la droite liant deux points de rendez-vous. La population va donc se balader le long de cette droite et statistiquement, la moitié dépassera et l'autre non. La sous-fonction d'évaluation de dépassement servira alors à éliminer ceux qui dépassent.

2.3.5 La viabilité

Il s'agit d'une sous-fonction d'évaluation binaire ou éliminatoire. Le modèle de notre avion est décrit par un point possédant trois angles (de lacet, roulis et tangage). Une autre hypothèse est qu'aucun de ses angles ne dépasse 30 degrés d'un point de trajectoire à un autre. Si un de ses trois angles dépasse les 30 degrés, l'individu est éliminé par cette sous-fonction d'évaluation.

Certaines sous-fonctions d'évaluation sont utilisées pour atteindre cet objectif de viabilité et éviter l'élimination de toute la population. Par exemple, la sous-fonction d'évaluation de l'évolution de la latitude versus la longitude rend stable le lacet de l'appareil tel expliqué dans la section 2.3.3.

2.3.6 Angle horizontal d'arrivée

Nous avons remarqué que l'algorithme a de la difficulté à l'approche des points de rendez-vous, plus particulièrement sur les contraintes de lacet (l'angle de lacet inférieur à 30 degrés). La trajectoire demandée à l'avion peut dépasser des courbes avec 30 degrés sur le plan horizontal, mais l'avion doit y parvenir avec des angles successifs de 30 degrés maximums. Il doit donc décomposer un mouvement supérieur à 30 degrés en plusieurs mouvements

inférieurs à 30 degrés. Mais sans l'utilisation d'une sous-fonction d'évaluation pour contrôler ce mouvement, il existe une saturation du système à l'approche des points de rendez-vous qui pourrait éliminer toute la population.

On a donc introduit une sous-fonction d'évaluation d'approche pour atteindre progressivement les exigences en lacet de chaque point de rendez-vous. Cette sous-fonction d'évaluation utilise l'angle entre la droite de trajectoire demandée par les deux points de rendez-vous joignant les points *A* et *B*, et la droite entre le point de rendez-vous à atteindre *B* et l'individu testé *A'* (figure 2.15).

Cette nouvelle sous-fonction d'évaluation permet de demander à l'algorithme de prévoir le virage demandé par chaque point de rendez-vous en avance et de calculer une trajectoire plus adaptée aux approches de ces points de rendez-vous.

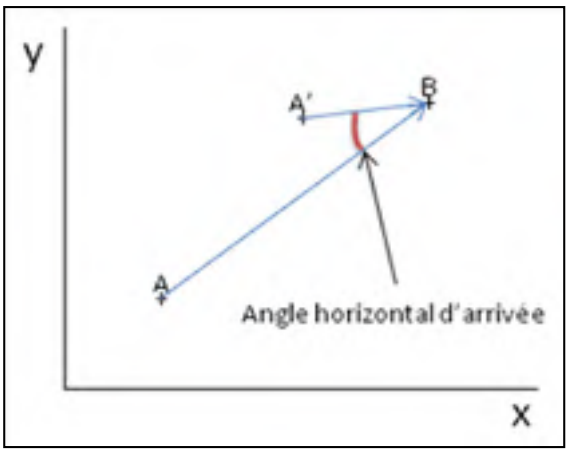

Figure 2.15 Fonction d'évaluation de l'angle horizontal d'arrivée.

2.3.7 Liaison des sous-fonctions d'évaluation

La liaison entre toutes ces sous-fonctions d'évaluation nous permet de calculer la fonction d'évaluation du système et ce calcul est réalisé en plusieurs étapes. Premièrement, tous les individus ne remplissant pas les sous-fonctions d'évaluation éliminatoires sont éliminées.

Deuxièmement, toutes les sous-fonctions d'évaluation sont normalisées entre 0 et 1, en utilisant l'équation (2.6). Troisièmement, les sous-fonctions d'évaluation sont rassemblées en une fonction d'évaluation unique pour chaque individu de la population.

Soit X_n la valeur normalisée, x la valeur en mètres, X_{min} la valeur minimale de la population en mètre et X_{max} la valeur maximale de la population en mètres, on obtient

$$X_n = (x - X_{min})/(X_{max} - X_{min}) \tag{2.6}$$

Chaque sous-fonction d'évaluation a son importance et n'a donc pas le même poids dans cette liaison, c'est pour cette raison qu'un coefficient fixe a été ajusté pour chacune. Dans l'équation de liaison (2.7), un coefficient de 1 pour une sous-fonction d'évaluation indique une implication normale et non valorisée. Une étude approfondie du poids de chaque sous-fonction d'évaluation a été faite dans les sous sections précédentes. Cette étude nous a permis de donner un poids à chaque fonction d'évaluation de manière expérimentale (tests sur chaque sous-fonction d'évaluation) pour obtenir les résultats de calculs de trajectoires. C'est l'utilisation et les essais indépendants de chaque sous-fonction d'évaluation qui ont permis d'obtenir l'équation (2.7) dans laquelle notre de notre fonction d'évaluation est calculée.

Soit F la fonction d'évaluation totale, $f1$ la sous-fonction d'évaluation euclidienne, $f2$ la sous-fonction d'ajustement en altitude, $f3$ la sous-fonction d'évolution latitude/longitude, $f4$ la sous-fonction de l'angle d'arrivée, et sont toutes normalisées entre 0 et 1. De plus, $f5$ est la sous-fonction d'évaluation de viabilité, et $f6$ est la sous-fonction d'évaluation de dépassement, alors on obtient,

$$\begin{cases} F = (0.1 * f1 + 0.5 * f2 + 1 * f3 + 2.4 * f4)/4 \\ F = 0, \text{si } f5 = 0 \\ F = 0, \text{si } f6 = 0 \end{cases} \tag{2.7}$$

En testant toutes les sous-fonctions fonctions d'évaluation et en les rassemblant nous arrivons à des résultats semblables montrés dans la figure 2.16. Ces résultats seront plus détaillés dans

le chapitre 3. Dans ces résultats, chaque point est un compromis donné par l'équation (2.7), de toutes les sous-fonctions d'évaluation et de toutes les contraintes énoncées dans les sections 2.3.1 à 2.3.7.

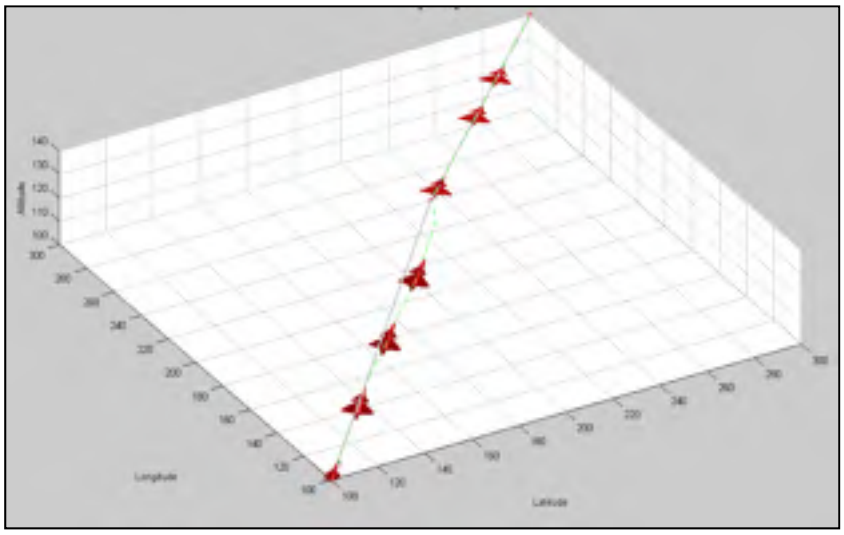

Figure 2.16 Trajectoire avec la fonction d'évaluation totale.

2.3.8 Le scaling

Le *scaling* [2] vise à recalculer la fonction d'évaluation d'un individu en prenant en compte la fonction d'évaluation moyenne de la population. Le but de cette manœuvre est d'éviter qu'un individu qui aurait la fonction d'évaluation très élevée, ne devienne trop prépondérant dans une population. Ceci aurait pour effet de détruire l'hétérogénéité de la population et de former des populations à un seul individu. Grâce à cette technique l'échelle diminue artificiellement et laisse une chance presqu'à tous les individus, incluant les moins adaptés.

Il existe deux types de *scaling* : linéaire et exponentiel [2]. Le *scaling* linéaire consiste à multiplier toutes les fonctions d'évaluations par un coefficient compris entre 0 et 1. De cette manière on réduit les espaces entre tous les individus. Cependant ce *scaling* devient handicapant en fin de recherche, lorsque le processus doit s'accélérer pour fournir la bonne solution. Ce désavantage n'est pas présent dans le *scaling* exponentiel. Celui-ci est représenté

parles équations (8) et (9), expliquées en détails prochainement. Pour le *scaling* exponentiel, le facteur *k* varie en fonction du nombre de générations *N*, alors *k* pourrait avoir les valeurs suivantes :

- Si $k \approx 0$, tous les écarts entre les fonctions d'évaluation sont réduits, et finalement il n'y a pas de différence entre les individus, qu'ils soient bons ou mauvais.
- Si $k \approx 1$, le scaling n'agit plus et aucune modification n'est apportée aux fonctions d'évaluation.
- Si $k > 1$, les écarts sont exagérés et les individus forts sont sélectionnés au détriment des plus faibles dans des proportions plus importantes que normalement.

Soit *F'* la fonction d'évaluation obtenue avec le *scaling* exponentiel, dans laquelle *F* est la fonction d'évaluation, *n* est le nombre de générations actuelles, et *k(n)* est le facteur exponentiel donné par l'équation (9), on obtient,

$$F' = F^{k(n)} \tag{2.8}$$

Soit *n* le nombre de la génération actuelle, *N* le nombre total de générations et *k* le facteur exponentiel, on obtient,

$$k(n) = \left(\tan \frac{\pi \alpha}{2(N+1)} \right)^{N-n} \tag{2.9}$$

Notre algorithme génétique possède un *scaling* exponentiel. Comme le montre les tableaux 2.5 et 2.6, celui-ci permet d'éviter les populations à un individu dans l'évolution des générations. Nous avons repris le cas où le point de départ est *A*(100,100,100) et le points d'arrivée est *B*(200,190,120). Nous avons figé l'algorithme avec une population de dix individus après 2 générations calculées et nous avons regardé les altitudes générées. Ce que le *scaling* nous montre est une évolution équilibrée sans prédominance d'une seule solution sous forme d'altitude (tableau 2.6), et c'est ce que l'on recherche. Lorsque le *scaling* est

appliqué, la population comporte des altitudes bien plus distribuées dans l'espace des solutions.

Tableau 2.5 Population sans *scaling* après 2 générations

Individus	Altitude (m)
1	109
2	109
3	109
4	109
5	108
6	107
7	108
8	109
9	107
10	108

Tableau 2.6 Population avec *scaling* après 2 générations

Individus	Altitude (m)
1	102
2	109
3	106
4	108
5	108
6	109
7	108
8	109
9	109
10	105

2.3.9 *Sharing* et *Clustering*

Le *sharing* (ou le partage) est un autre moyen de contrôler la fonction d'évaluation d'un individu. En effet cette méthode permet de combattre le regroupement afin de conserver un bon brassage génétique (figure 2.17). Le brassage génétique consiste dans la dispersion, dans l'espace des solutions, des individus de la population pour éviter de se retrouver avec une majorité d'une population formée d'individus trop proches et semblables (voir la figure 2.17). En d'autres termes, un bon brassage génétique amène une population hétérogène.

Techniquement le regroupement de plusieurs individus dans le même espace de solution déclenche le *sharing* qui dégrade leurs fonctions d'évaluation. On calcule donc la distance entre les individus et si celle-ci est inférieure à une distance spécifiée par l'utilisateur de la méthode, alors la fonction d'évaluation est dégradée. Le problème d'une telle méthode est le temps de calcul. En effet il faut évaluer la distance entre chaque individu et le reste de la population.

Afin de palier au problème du temps de calcul, les chromosomes sont rassemblés en *clusters* (ou tableaux). On les regroupe pour réduire le temps de calcul. Un *cluster* est un groupement d'individus et il est identifié par un centre et une amplitude. Le rassemblement est effectué de la manière suivante: le premier individu de la population devient le centre du premier cluster et ensuite on le compare ensuite avec les autres individus. Si la distance entre ce centre et un autre individu est inférieure à l'amplitude du *cluster*, cet individu appartient désormais au cluster. Si cette même distance est plus grande que l'amplitude, l'individu considéré devient un nouveau cluster. Lors de l'addition d'un individu à un *cluster*, on recalcule son centre en fonction des individus déjà présent et du nouvel arrivant.

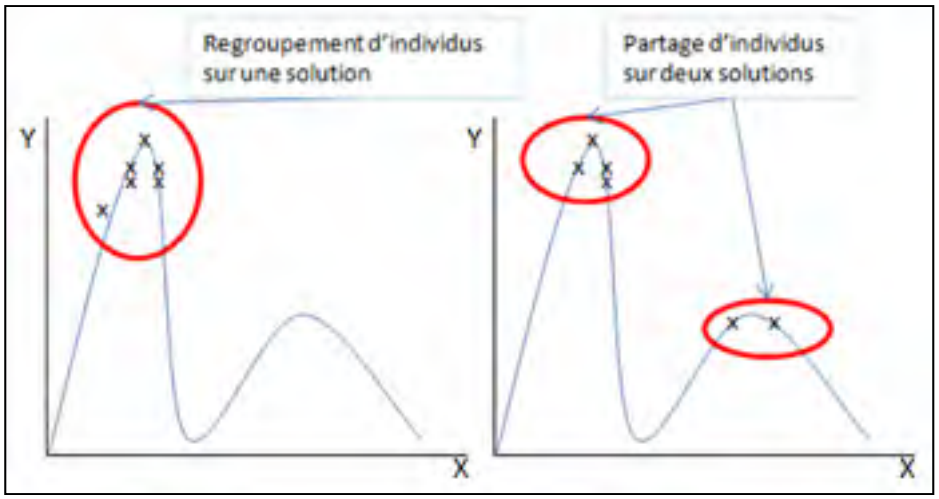

Figure 2.17 À gauche sans sharing, concentration des individus sur une seule solution À droite avec sharing, partage des individus sur les deux solutions.

Le *sharing* n'a pas été utilisé dans notre projet. En effet, le brassage génétique était déjà assuré par d'autres méthodes comme le *scaling*, et le sharing aurait alourdi l'algorithme (les temps de calcul étant déjà élevés). De plus les résultats obtenus suffisaient et un alourdissement du code n'aurait pas été optimisé au niveau du temps de calcul et de la pertinence des résultats.

2.4 Le croisement

Afin de garder une population homogène et de ne pas favoriser les individus ayant une bonne fonction d'évaluation, il faut brasser génétiquement les populations enfants grâces aux opérateurs de croisement et de mutation. En d'autres termes le croisement vise à générer une nouvelle population d'individus à partir de la population actuelle afin d'obtenir la solution désirée. Le croisement consiste à accoupler deux individus pour obtenir deux individus enfants. C'est ainsi que l'on construit la génération suivante. Celle-ci doit être plus adaptée aux problèmes et contraintes posées par ces problèmes. Avant d'accoupler des individus, il faut les sélectionner dans la population actuelle et comme méthode de sélection, nous avons

choisi un croisement par sélection proportionnelle à l'adaptation. Voici les différents choix qui s'offraient à nous :

- *Sélection uniforme*. On sélectionne les individus à accoupler de façon aléatoire. Chaque individu a donc autant de chances que les autres d'être sélectionné, avec une probabilité de $1/N$, si N est le nombre d'individus de la population parents.

- *Sélection par tournoi*. Ici on associe par paires les individus et on détermine ensuite directement qui a le meilleur score d'adaptation. Celui-ci sera choisi pour le croisement. En d'autres termes, la sélection s'effectue par paires et non individuellement.

- *Sélection par rang*. On choisit et on accouple les individus les plus forts. Il n'y a donc pas de hasard. Les plus faibles (faible fonction d'évaluation) sont éliminés.

- *Sélection proportionnelle à l'adaptation*. La faculté d'adaptation de l'individu face à un problème (fonction d'évaluation) détermine sa probabilité d'être choisi. On a donc une roue de fortune biaisée qui avantage les individus ayant des bonnes fonctions d'évaluation.

La sélection proportionnelle à l'adaptation utilise la loi du plus fort mais permet également une part de hasard proposant aux individus ayant une faible fonction d'évaluation une faible chance d'être sélectionnées. Le processus ne les élimine donc pas. En effet, bien que des individus mal adaptés aient peu de chance d'être choisis, il reste possible qu'ils puissent s'accoupler et qu'ils restent dans la génération suivante. Il est possible qu'ils possèdent des caractéristiques adaptées aux contraintes de la solution recherchée mais que celles-ci ne s'expriment pas avec eux comme elles le feraient avec leur enfant (Voir exemple du paragraphe suivant).

Par exemple, si on a une altitude de 100 mètres et que nous désirons atteindre 300 mètres. Voici une liste d'individus que nous possédons pour notre population : 186, 203, 245. A priori les deux derniers sont les meilleurs car ils se rapprochent plus de l'objectif (300 m).

Le croisement consiste dans le nombre de 186 (choisi arbitrairement) que nous allons découper en deux. On a donc une partie 1 et une partie 86. On fait de même avec les autre nombre et on les croise entre eux. Dans le cas d'un croisement entre 186 et 203 on peut avoir comme enfant 286 et 103. Le nombre 286 possède le 2 de 203 et le 86 de 186, alors 286 est le meilleur enfant que cette population de 3 membres auraient pu nous donner, et il est le plus proche de 300. Il ne fallait donc pas l'écarter et la sélection proportionnelle à l'adaptation permet ceci. (Ici 186 et 245 donne 286 et 145 et de même 203 et 245 donne 205 et 243).

Pour la sélection proportionnelle à l'adaptation [2], on crée donc une roue de fortune baisée dans laquelle chaque individu possède une partie proportionnelle à sa fonction d'évaluation, comme le montre la figure 2.18. On tire ensuite aléatoirement les individus dans cette roue ou les plus forts ont plus de chance d'être sélectionnés et les plus faibles existent tout de même. La figure 2.18 est un exemple pour une population à 5 individus (choix arbitraire) où l'individu 1 est le plus fort et l'individu 2 est le plus faible.

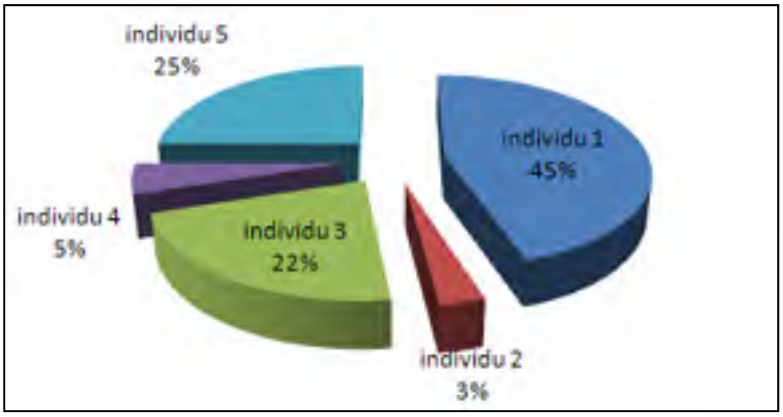

Figure 2.18 Exemple de roue de fortune biaisée.

Une fois les individus sélectionnés, il faut les croiser deux-à-deux. Le croisement de deux individus parents donne deux individus enfants. Le croisement a plusieurs choix pour son implémentation. On peut le réaliser en nombres binaires ou en nombres décimaux.

L'avantage du croisement avec des nombres binaires est la grandeur des nombres qui possèdent plus de chiffres. Par exemple 1100 est plus grand que 12 en termes de nombre de chiffres. Pourtant 1100 et 12 représentent le même nombre. Le codage binaire permet la réalisation de plusieurs troncatures et donc des entrecroisements, permettant ainsi un brassage génétique plus grand, ceci risquant de ralentir l'algorithme. En effet plus il y a de brassage, plus il y a de hasard, perdant ainsi les individus les plus adaptés. Il en faut donc du brassage, mais pas à outrance. Un autre avantage du croisement binaire est la normalisation des données. En binaire, qui est un langage universel, toutes les données sont dans ce même langage et peuvent donc s'accoupler. Cependant nos données étant toutes des distances, cette normalisation en binaire n'est pas nécessaire. Voici un exemple de croisement sur deux individus binaires avec plusieurs entrecroisements :

- Étape 1 : Parents :

  Parent 1 : 10011001001

  Parent 2 : 01110110010

- Étape 2 : Localisation des croisements :

  Parent 1 : 10|011|0010|01

  Parent 2 : 01|110|1100|10

- Étape 3 : Enfants :

  Enfant 1 : 10|110|0010|10

  Enfant 2 : 01|011|1100|01

Pour notre implémentation, nous voulions obtenir rapidement une solution. De plus, la perte de trop d'individus adaptés n'était pas acceptable pour un parcours plus grand de l'espace de solutions. Il était donc préférable de cibler plus les solutions plutôt que de se disperser.

La sélection proportionnelle à l'adaptation était présente pour réaliser le brassage dont nous avions besoins (gardant tout de même les individus à faible fonction d'évaluation). Il a donc été choisi d'opérer avec un croisement décimal et une seule troncature. Afin d'ajouter encore plus de brassage et permettre au hasard d'avoir également sa part, le lieu de la troncature reste aléatoire. Voici un exemple de troncature effectuée dans notre programme :

- Étape 1 : Parents :

  Parent 1 : 186

  Parent 2 : 203

- Étape 2 : Localisation des croisements :

  Parent 1 : 1|86

  Parent 2 : 2|03

- Étape 3 : Enfants :

  Enfant 1 : 2|86

  Enfant 2 : 1|03

Le programme ainsi implémenté possède également une sécurité. En effet un croisement pourrait amener un individu qui dépasse l'espace des solutions admissibles. Il faut donc éviter qu'il soit gardé dans la génération naissante. Un tel individu est supprimé dès sa naissance.

2.5 La mutation

L'opérateur de mutation (figure 2.19) est présent pour s'assurer que tout l'espace des solutions est parcouru. Il sert également à empêcher l'algorithme de stagner sur une solution en s'aveuglant à toute autre. C'est un opérateur totalement aléatoire. On prend un bit au hasard dans un individu et on le remplace par un autre bit générée également au hasard. En toute rigueur et théoriquement, si l'on n'appliquait pas l'opérateur de croisement, la mutation suffirait à long terme pour couvrir tout l'espace des solutions. Ceci en fait un opérateur primordial, mais, par contre, il implique beaucoup de dispersion dans la recherche. Il faut donc le limiter, et en même temps lui garantir sa place. En d'autres termes, pour notre problématique chaque bit de chaque individu a une probabilité de muter qui est le taux de mutation. Dans notre cas cette probabilité vaut 0.001. Nous avons testé des taux de mutations plus élevé que 0.001, mais la génération de la population enfant devenait trop aléatoire et les trajectoires générées devenaient trop chaotiques.

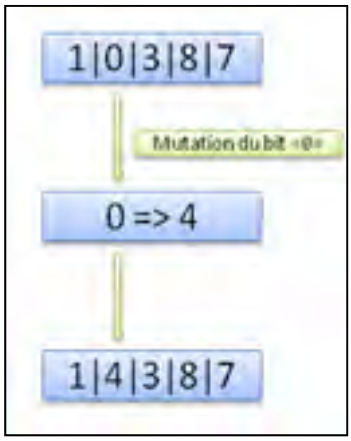

Figure 2.19 Exemple de mutation.

L'opérateur de mutation possède également une sécurité, comme l'opérateur de croisement pour éviter de dépasser l'espace des solutions. Cette méthode de mutation simple fut choisie à l'égard de principes de mutations plus compliqués, évoluant en fonction du nombre d'itérations de l'algorithme. Par exemple, la mutation adaptative se base sur un taux de mutation variant et codé dans la structure de nos individus. Elle fait partie de chaque individu

et subit également les croisements et les mutations (de son propre taux de mutation actuel à chaque itération). Le taux de mutation dégénère avec le nombre d'itérations. Plus on se rapproche de la solution optimale, plus le taux de mutation est faible, handicapant ainsi le brassage génétique (ou la dispersion des solutions). C'est pourquoi dans cet algorithme, la méthode de mutation à taux variable n'a pas été retenue, face à un taux de mutation constant durant tout l'algorithme.

2.6 L'élitisme

À la création d'une nouvelle population, le ou les meilleurs individus seront perdus après les opérations de croisement et de mutation. Pour éviter ceci, on utilise la méthode de l'élitisme (figure 2.20). Cette méthode consiste à directement copier un ou plusieurs des meilleurs individus de la génération parents vers la génération enfants. Ensuite, on génère le reste de la population selon l'algorithme habituel. Cette méthode améliore considérablement les algorithmes génétiques, car elle ne permet pas la perte de la meilleure solution (dans notre cas nous n'en gardons qu'une solution pour améliorer notre brassage génétique et ne pas former une sous-population d'élite) tout en continuant à effectuer le brassage génétique et la recherche de la solution optimale.

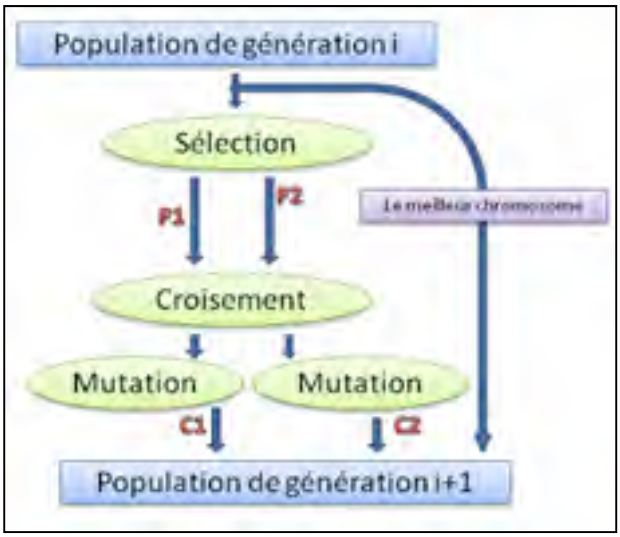

Figure 2.20 Principe de l'élitisme.

Afin de montrer l'efficacité de l'élitisme, nous avons comparé les mouvements de roulis de trajectoires générées avec et sans élitisme. Ces résultats sont montrés dans les figures 2.21 et 2.22. Le roulis est en radians, sachant que 30 degrés équivaut à 0.56 rad.

Dans les figures suivantes 2.21 et 2.22, on peut voir que l'élitisme offre une trajectoire plus stable et continue en roulis. Bien que le cas sans élitisme ne donne finalement que des solutions stables, celui-ci reste néanmoins extrêmement bruité sur le début. Ceci est du au fait que chaque point calculé n'est pas forcément le meilleur, ce qui n'est pas le cas avec élitisme.

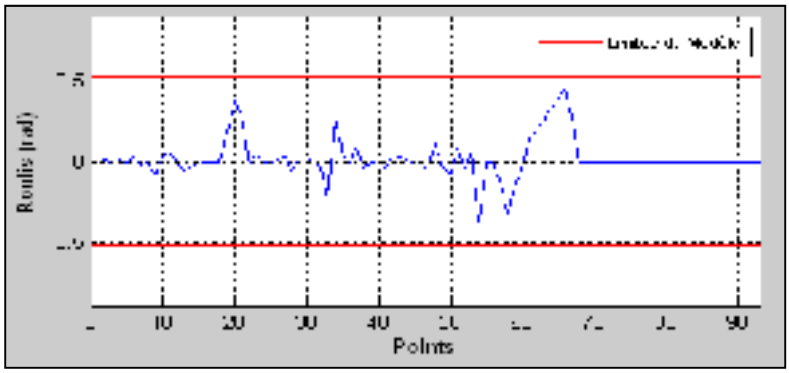

Figure 2.21 Roulis versus le point de trajectoire sans élitisme.

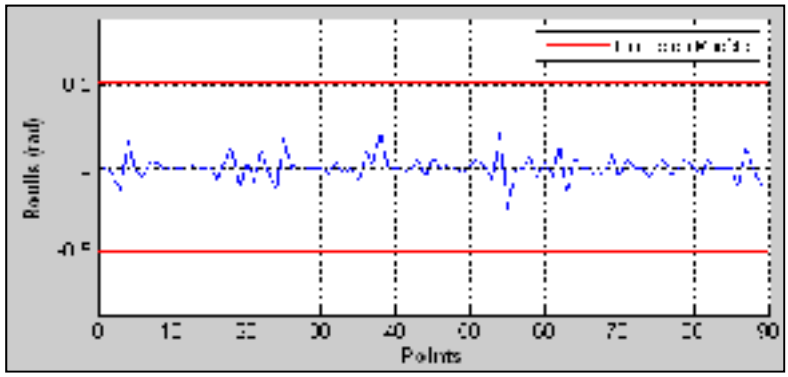

Figure 2.22 Roulis versus le point de trajectoire avec élitisme.

2.7 Le recuit simulé

Le recuit simulé est une autre méthode changeant radicalement l'algorithme et visant à garder les meilleurs éléments de notre population. Elle n'a pas été utilisée car nous avons préféré l'élitisme, cependant il nous semble nécessaire d'en parler et de donner nos conclusions. Le recuit simulé se produit pendant le croisement. Une fois que deux individus parents sont croisés pour former deux individus enfants, on évalue et compare les deux parents avec leurs deux enfants afin de garder les deux meilleurs individus de tous (figure 2.23). On ne perd ainsi pas d'individus mieux adaptés à notre problème d'une génération à une autre.

Comparons l'élitisme et le recuit simulé. L'élitisme ne concerne qu'un seul membre de la population : le meilleur, et se produit une fois par génération. Le recuit simulé, lui, s'effectue à chaque croisement. Il n'y a donc moins de risque avec l'élitisme, de création d'une population élite qui se partage tous les croisements. Avec le recuit simulé il y a des risques de voir ce genre d'élite se former. C'est pourquoi notre choix s'est porté sur l'élitisme.

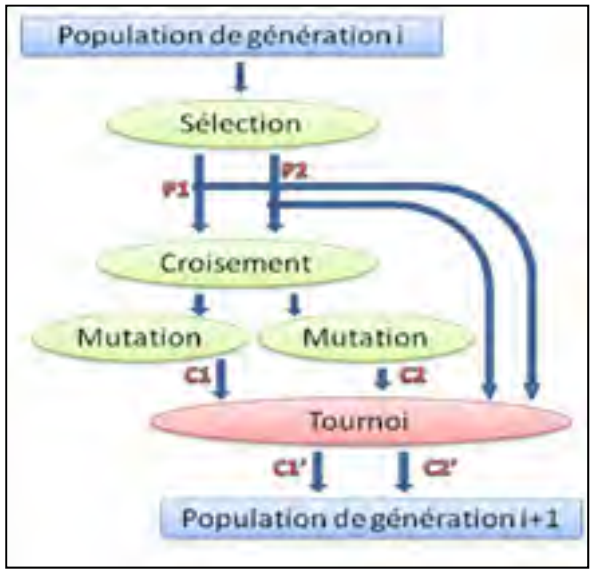

Figure 2.23 Principe du recuit simulé.

2.8 Les critères d'arrêt et la division du problème

L'arrêt de l'algorithme se traduit généralement par deux mécanismes : le nombre d'itérations et la proximité d'une solution désirée. Le nombre d'itérations détermine le nombre maximal de générations voulues. Une fois atteint, l'algorithme s'arrête et son meilleur résultat est la trajectoire affichée pour ce nombre de générations. Dans le cas du deuxième mécanisme, c'est-à-dire, dans le cas de la proximité d'une solution désirée, une fois que l'algorithme trouve une solution assez proche de la solution optimale, il s'arrête.

Dans notre cas, nous avons utilisé les deux mécanismes. En effet, la structure de notre algorithme génétique nécessitait ces deux mécanismes. Le calcul de trajectoire que l'algorithme devait effectuer était trop long pour se faire d'un coup. Nous avons donc décidé de diviser le problème, et donc la trajectoire à calculer, et d'appliquer les *AG* à chacune de ces sous-parties.

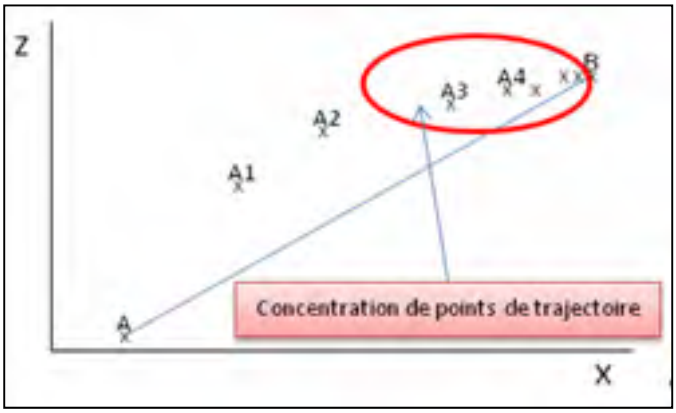

Figure 2.24 Risque de trajectoire calculée sans division du problème.

La figure 2.24 nous montre les résultats attendus si le problème n'est pas divisé. Les *AG* convergent lentement vers une solution une fois qu'ils l'approchent. On risque d'obtenir un nuage de points à l'approche des points de rendez-vous, ainsi que peu de points au lancement de l'algorithme.

Ce genre de résultats n'est pas acceptable pour plusieurs raisons. Premièrement, on nous demande une trajectoire homogène dans l'espace. En aucun cas on ne doit avoir des nuages de points. Deuxièmement le programme créera obligatoirement un échec de calcul, car dans les nuages, les points possèdent des dérivées presque infinies. En effet, les points sont alors très proches les uns des autres. Or ces dérivées représentent le lacet, le roulis et le tangage. Le programme éliminera donc toute la population à l'approche des points de rendez-vous.

Pour palier à ce problème, il a été décidé de diviser la trajectoire par des périodes d'itérations (figure 2.25). En d'autres termes, on calcule la trajectoire pendant un petit nombre d'itérations, et ensuite si la solution n'a pas été trouvée, on relance l'algorithme à partir du dernier point trouvé. De cette manière la trajectoire est homogène et on évite les nuages de points montrés dans la figure 2.24 à proximité du point *B*.

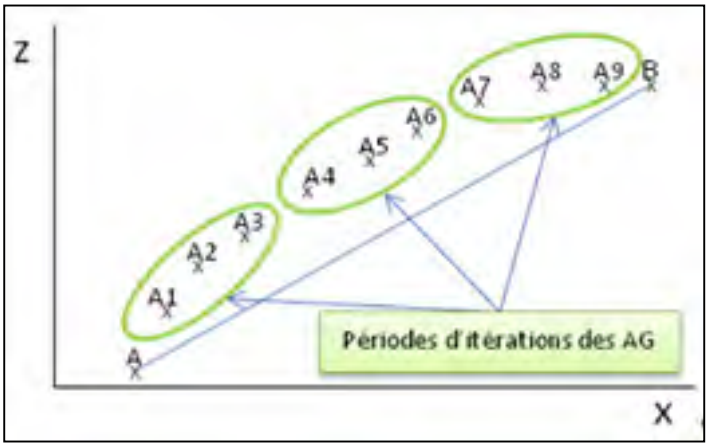

Figure 2.25 Trajectoire calculée en tenant compte des périodes d'itérations.

Il ne faut pas oublier que nous possédons tous les points de rendez-vous. Le but des *AG* n'est donc pas uniquement de les retrouver mais de bien les retrouver. C'est donc l'évolution des *AG* vers nos points de rendez-vous que l'on veut contrôler.

La proximité d'une solution est calculée simplement par la différence de deux normes. Il s'agit de savoir si le dernier point trouvé a atteint un point de rendez-vous et donc si l'on peut passer au point de rendez-vous suivant. La première norme est calculée entre le point de

rendez-vous et le dernier point trouvé (qui est aussi testé). La deuxième norme est calculée entre les deux points de rendez-vous entre lesquels notre algorithme évolue, divisée par 100. Ceci équivaut à dire que dès que l'avion est à 99% de la solution, c'est qu'il l'a atteint.

CHAPITRE 3

RESULTATS

3.1 Introduction

Cette section se consacre aux résultats et à leurs significations. Nous allons en premier lieu présenter les résultats finaux utilisant notre algorithme génétique sur une trajectoire de 5 points de rendez-vous. Nous effectuerons ensuite la liste des différents paramètres que nous pouvons varier et de leurs effets sur l'évolution de notre trajectoire. Ces paramètres sont les suivants :

- Le nombre d'itérations des sous-périodes énoncées dans la section 2.8. Ici il sera question de la division du chemin à parcourir. En d'autres termes, nous sectionnerons de manière plus ou moins importante le chemin à parcourir par l'algorithme afin d'analyser les conséquences sur le calcul final de la trajectoire.
- Le taux de mutation.
- Le nombre de chromosomes ou d'individus.
- Le taux de rapprochement des points de rendez-vous entre le point donné par l'utilisateur et le point recalculé par notre programme tel vu dans la section 1.4.
- Les conditions initiales de notre algorithme génétique.

Dans le tableau 3.1 nous indiquons les coordonnées des cinq points de rendez-vous qui forment la trajectoire à calculer. Ces cinq points seront constants dans l'étude afin de comparer les résultats obtenus. Ces points seront choisis selon plusieurs critères.

On voulait que l'un de ces points (x = la latitude) possède une évolution constante de 100 mètre (tableau 3.1) pour confirmer que notre algorithme possèderait une évolution constante (sans accélération de la vitesse de calcul). Il fallait qu'un deuxième paramètre (y = longitude) possède des décélérations et des accélérations (montrées dans le tableau 3.1) dans son évolution, pour vérifier la capacité de notre algorithme à répondre à ces accélérations et décélérations. Enfin, le troisième paramètre (l'altitude) devrait progresser pour ensuite

revenir aux valeurs de départs (tableau 3.1) pour vérifier la capacité de notre algorithme faire de même.

Tableau 3.1 Coordonnées des cinq points de rendez-vous

Numéro du point	Latitude (m)	Longitude (m)	Altitude (m)
1	100	100	100
2	200	190	120
3	300	250	140
4	400	300	110
5	500	500	100

3.2 La réponse et la validation de l'algorithme

La réponse représente le cas calculé avec les paramètres optimaux pour répondre à notre problème. C'est donc le résultat obtenu par notre algorithme génétique en tenant compte des spécifications qui regroupent :

- un nombre d'itérations des sous-périodes de 10;
- un taux de mutation de 0.001;
- un nombre de chromosomes de 400;
- un taux de proximité des points de rendez-vous de 85%;
- des conditions initiales n'avantageant aucune des données du tableau 3.1.

Les résultats sont présentés dans les figures 3.1 à 3.9. Ces figures regroupent les six dimensions de l'avion. Elles regroupent également la trajectoire en trois dimensions et la route horizontale ainsi qu'une étude approfondie de l'évolution de l'altitude de l'avion. Si l'on regarde en détails la trajectoire et la route horizontale de l'avion (figures 3.1 et 3.2), on peut remarquer que celles-ci répondent à nos attentes. La trajectoire suit les points de rendez-vous d'une manière intelligente selon les spécifications données ce qui permet le calcul d'une trajectoire optimale. Les figures 3.3 et 3.4 décrivant la latitude et la longitude nous confirment la volonté de suivre les points de rendez-vous recalculés. On remarque qu'il n'y a

pas de dépassement de trajectoires et que les trajectoires sont fluides aux niveaux des points de rendez-vous. Chaque sous-fonction d'évaluation (voir sous-sections 2.3.1 à 2.3.7), et donc contrainte appliquée à l'algorithme, semble remplir son rôle, et ainsi leur fusion permet d'obtenir des résultats qui répondent à notre problématique.

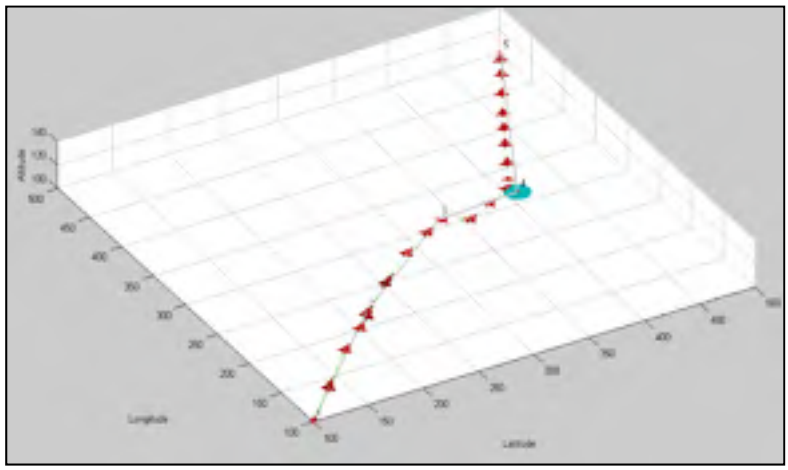

Figure 3.1 Trajectoire globale suivant la latitude, la longitude et l'altitude.

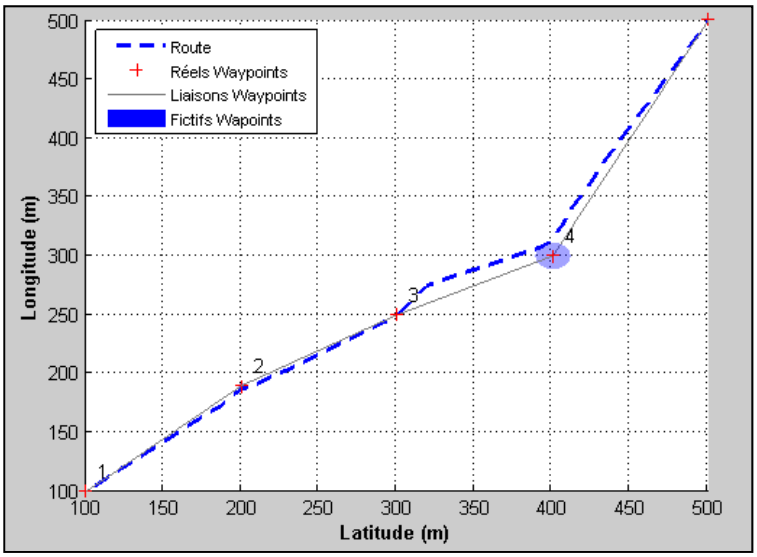

Figure 3.2 Route horizontale de notre avion (longitude versus latitude).

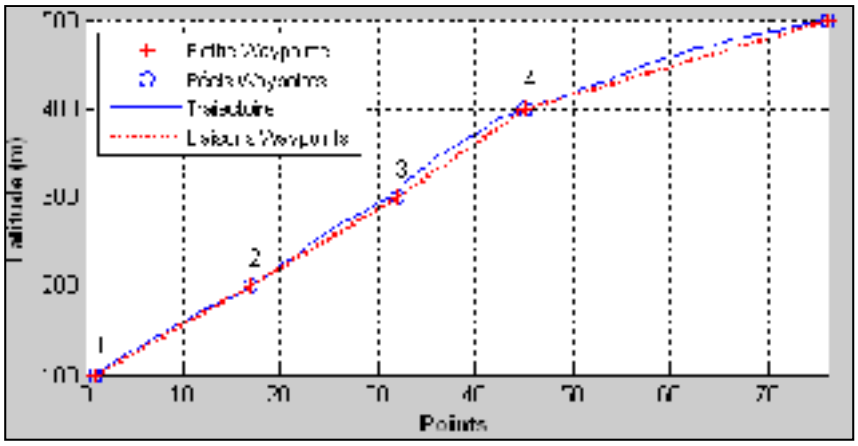

Figure 3.3 Latitude versus le point de trajectoire.

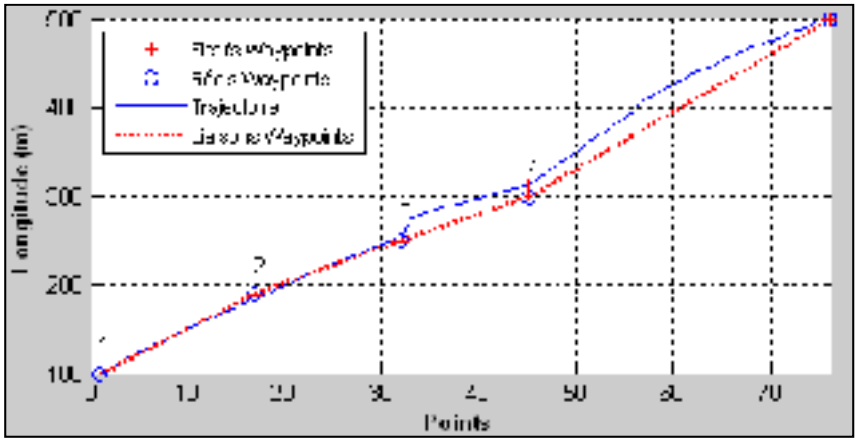

Figure 3.4 Longitude versus le point de trajectoire.

Les évolutions de l'altitude et du tangage en fonction des points de trajectoires sont détaillées dans les figures 3.5 et 3.6. En effet, leur but n'est pas de suivre la droite tracée entre chaque point de rendez-vous mais de rejoindre au plus vite, en respectant les contraintes de l'avion, l'altitude du point de rendez-vous suivant, tel montré dans la figure 3.5. Comme on peut le voir, une telle évolution est calculée par notre algorithme en gardant une bonne stabilité en tangage. Les paliers de tangage montrés dans la figure 3.6 possèdent des variations assez faibles, sauf pour rejoindre le point 4 ou le tangage demandé est très élevé, ce qui demande

des corrections de tangages plus serrées à l'algorithme. Les seuls endroits ou le tangage est nul c'est lorsque l'on a atteint l'altitude d'un des points de rendez-vous suivant.

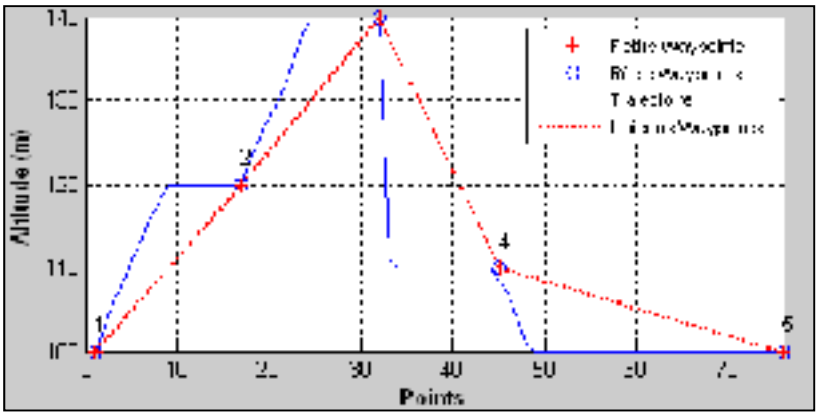

Figure 3.5 Altitude versus le point de trajectoire.

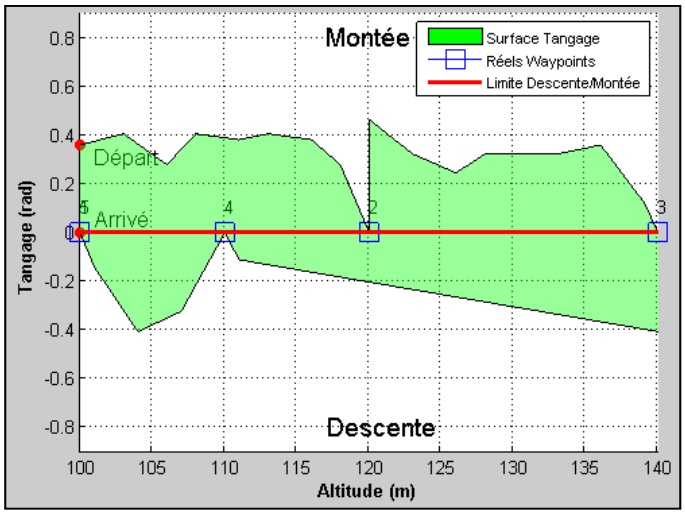

Figure 3.6 Évolution de l'altitude versus le tangage.

Les figures 3.7, 3.8 et 3.9 nous montrent les évolutions en lacet, roulis et tangage de l'avion. Les premiers phénomènes que l'on remarque, sont les périodes de fluctuations régulières et anormales présentes dans le lacet et le roulis (figures 3.7 et 3.8). Elles sont dues au passage d'un point de rendez-vous à un autre. En effet, l'avion se déplace sur des lignes droites entre

les points de rendez-vous, et il n'y a pas besoin de considérer les variations de lacet et roulis durant ces phases. Mais lorsque l'on arrive à une jonction (point de rendez-vous), il faut répondre au passage tout en respectant les contraintes demandées. Ceci est répondu par l'algorithme avec un rapprochement sévère de la contrainte d'angle maximum de 30 degrés afin d'accéder le plus rapidement possible au nouvel état d'équilibre représenté par la portion suivante de la trajectoire. Le tangage qu'en à lui, se comporte comme décrit précédemment. L'élitisme de notre algorithme génétique garde toujours la meilleure solution, alors l'altitude atteinte sera gardée et le tangage restera à zéro une fois celui-ci atteint. En effet, l'algorithme gardera la meilleure solution quelque soit le brassage génétique.

Dans les figures 3.7, 3.8 et 3.9 les unités sont en radians et 30 degrés vaut 0.56 radians.

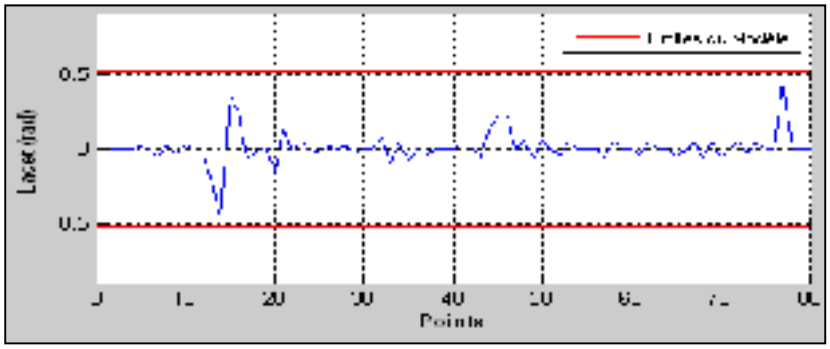

Figure 3.7 Lacet versus le point de trajectoire.

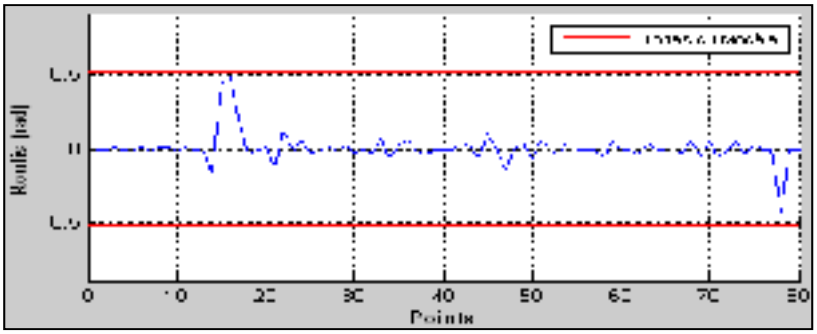

Figure 3.8 Roulis versus le point de trajectoire.

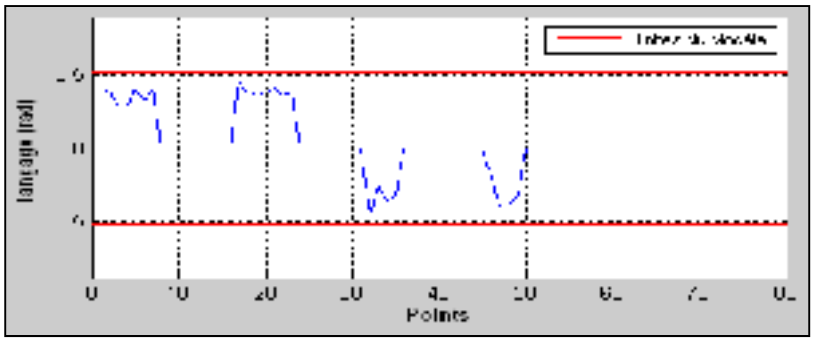

Figure 3.9 Tangage versus le point de trajectoire.

3.3 Analyse pour 10 et 600 chromosomes

Il s'agit ici de confirmer le choix de 400 pour le nombre de chromosomes. Pour cette configuration, nous avons procédé à une série de tests, dont nous allons vous montrer leurs limites maximums et minimums. Ces conditions regroupent :

- un nombre d'itérations des sous-périodes de 10;
- un taux de mutation de 0.001;
- **un nombre de chromosomes (ou individus) de 10 ou 600;**
- un taux d'approchement des points de rendez-vous de 85%;
- des conditions initiales n'avantageant aucune des données du tableau 3.1.

Les figures 3.10 à 3.14 nous montrent la série de résultats obtenus avec un nombre de chromosomes de 10. Il s'agit de démontrer que ce nombre est trop bas pour obtenir des résultats répondants aux spécifications. En effet une population de 10 chromosomes ne peut fournir d'individus répondant à toutes les spécifications. Les figures 3.10 et 3.11 nous montrent une trajectoire globale chaotique ayant une stabilité précaire et incertaine, et, une route horizontale. L'algorithme doit répondre à trop de contraintes pour une population de petite taille. Il est donc presque impossible que la population évolue si rapidement. Une solution pour remédier à ce problème, serait d'augmenter le nombre d'itérations des sous-périodes. Ce principe sera plus détaillé dans la section suivante 3.4 sur le nombre d'itérations dans les sous-périodes.

Si l'on prend en compte le nombre de contraintes (ou les sous-fonctions d'évaluation), il est nécessaire que la population couvre un espace maximum de solutions répondant à ses contraintes, sinon il lui sera facile de négliger des contraintes comme on peut le voir. La force des algorithmes génétiques se trouve dans le brassage génétique et l'un de ces pré-requis est l'abondance du nombre de chromosomes.

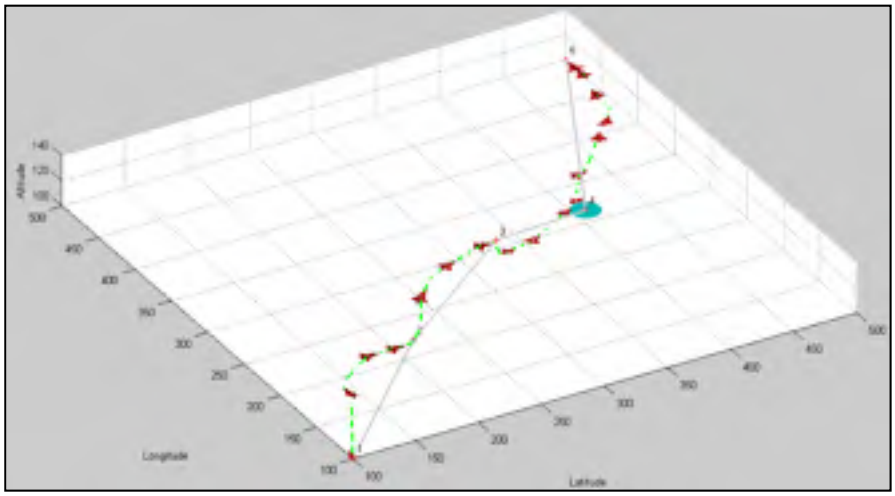

Figure 3.10 Trajectoire globale en trois dimensions pour 10 chromosomes.

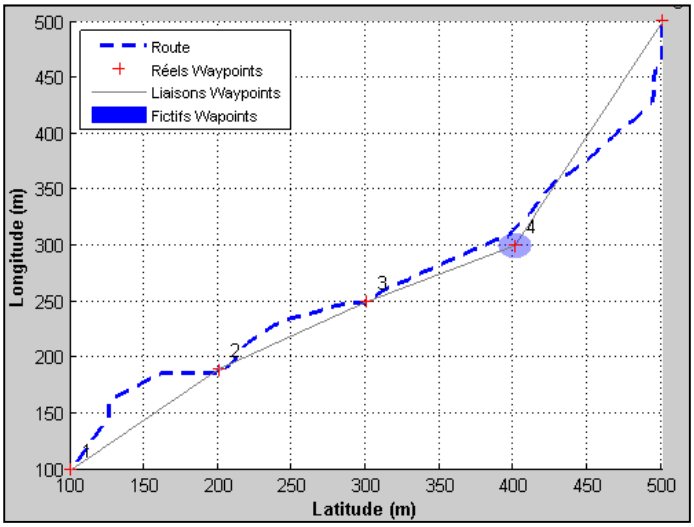

Figure 3.11 Route horizontale de notre avion en utilisant 10 chromosomes (longitude versus latitude).

La stabilité de l'avion étant précaire, des variations de l'angle de roulis assez importantes ont lieu (figure 3.12). Bien que les angles limites de 30 degrés soient maintenus, il n'en reste pas moins que le modèle est loin d'être stable. Les solutions acceptées restent les meilleures, mais dans une population de dix individus, qui possède donc une faible exploration de l'espace totale des solutions.

Dans la figure 3.13, nous exposons la variation de l'altitude avec les numéros des points. La première remarque est l'apparition de plusieurs paliers entre les points de rendez-vous 3 et 4, ce qui n'est pas une trajectoire acceptable et indique bien que le nombre de dix chromosomes est insuffisant. La figure 3.14 nous montre l'instabilité constante et les variations répétées de l'altitude et du tangage.

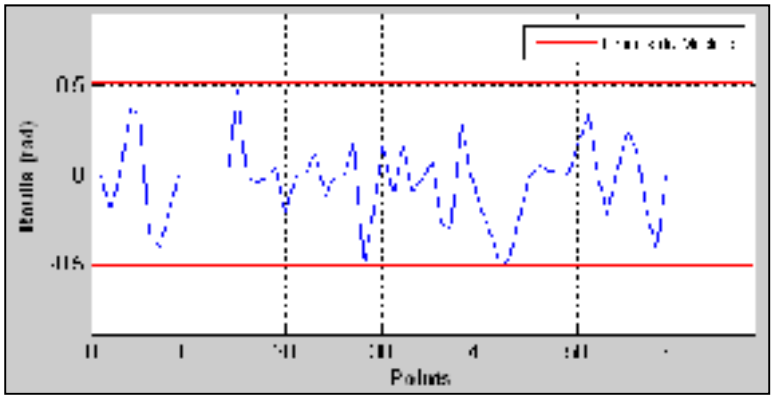

Figure 3.12 Roulis versus le point de trajectoire pour 10 chromosomes.

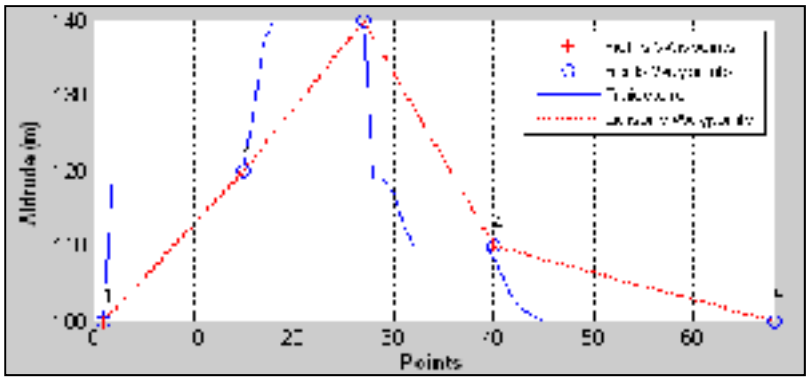

Figure 3.13 Altitude versus le point de trajectoire pour 10 chromosomes.

On peut remarquer des résultats intéressants si l'on compare l'évolution de l'altitude avec la route. Au démarrage, la solution privilégiée par l'algorithme est très liée à l'ajustement de l'altitude, et non le suivi de la route horizontale. Mais dès que les points s'éloignent trop de la route horizontale à suivre, l'algorithme dépriorise l'altitude et il se concentre à nouveau sur la route. On dirait qu'il passe d'une contrainte à une autre sans les satisfaire au même moment. Ceci confirme encore qu'il ne parcours pas assez l'espace des solutions et qu'il possède une grande instabilité sur ses réponses.

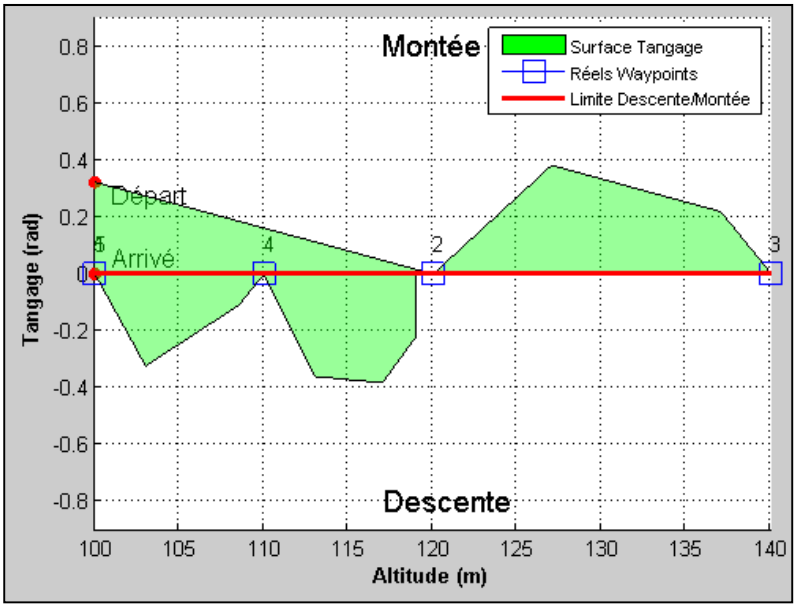

Figure 3.14 L'évolution de l'altitude versus le tangage pour 10 chromosomes.

Les figures 3.15 à 3.18 nous montrent la série de résultats obtenus avec 600 chromosomes. L'augmentation du nombre de chromosomes n'améliore pas la réponse de l'algorithme sur la solution désirée, mais augmente considérablement le temps de calcul. Pourtant les résultats sont assez impressionnants. Si l'on se penche sur la figure 3.15 représentante de la route horizontale, on remarque qu'elle colle presque parfaitement à la route que l'on désire. Plus loin, la figure 3.16, nous montre l'évolution de l'altitude. Celle-ci répond à la contrainte

d'ajustement mais plus loin, la figure 3.17 nous montre la stabilité de cet ajustement. Les variations de l'angle de tangage sont faibles et l'avion peut donc monter d'une bonne façon.

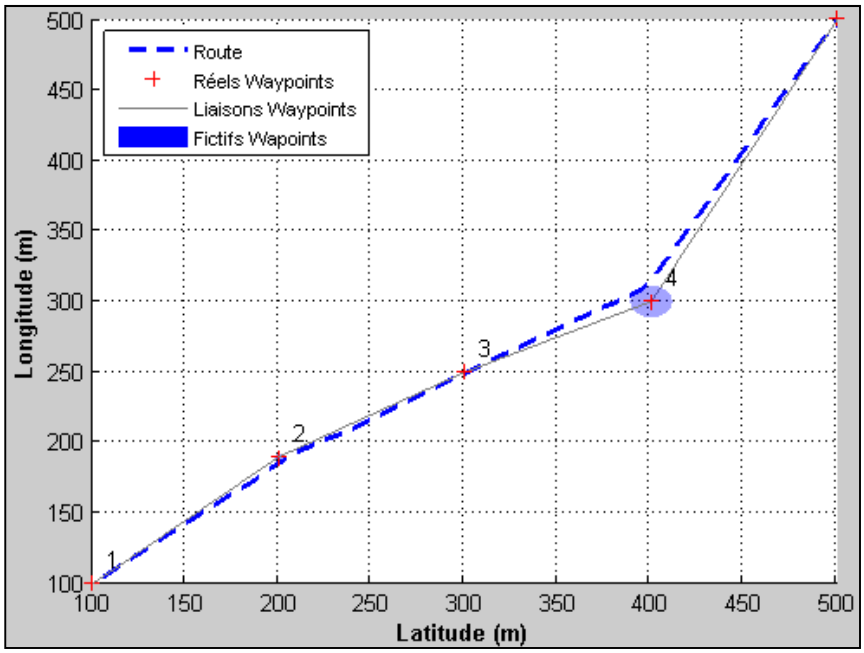

Figure 3.15 Route horizontale de l'avion avec 600 chromosomes (longitude versus latitude).

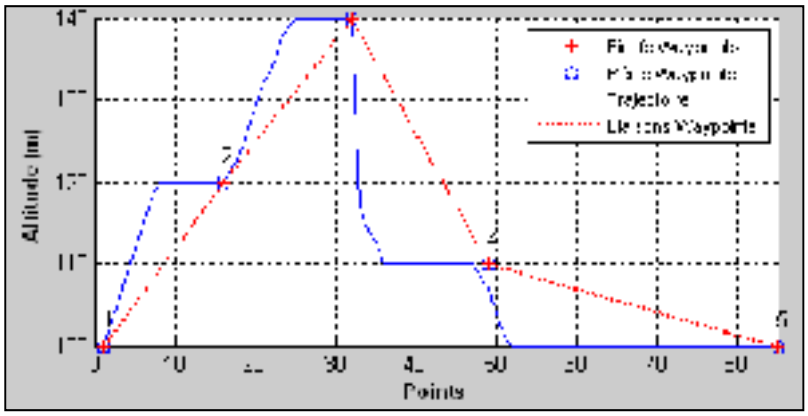

Figure 3.16 Altitude versus le point de trajectoire pour 600 chromosomes.

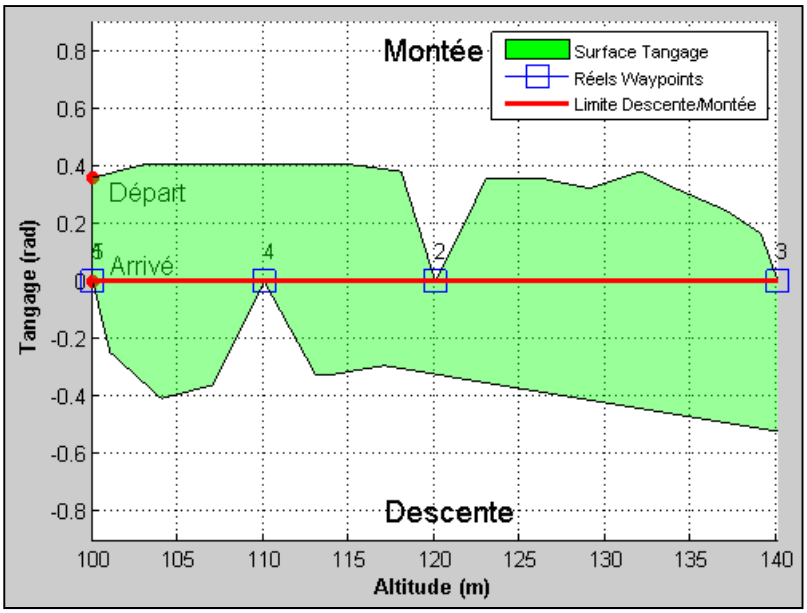

Figure 3.17 Évolution de l'altitude versus le tangage pour 600 chromosomes.

Enfin, la figure 3.18 nous expose les résultats obtenus pour l'angle de roulis pour 600 chromosomes. Mis à part la correction effectuée au niveau du point de rendez-vous 2, on constate la différence entre les résultats obtenus pour une population de 600 chromosomes par rapport à une population de 10 chromosomes. Les variations sont généralement faibles, inférieures à 10 degrés. Cette étude nous montre qu'il est nécessaire de trouver un nombre minimum de chromosomes pour le fonctionnement de l'algorithme, mais qu'un nombre démesuré n'est pas forcément nécessaire car la solution ne s'améliore pas significativement. La différence que l'on peut voir entre les résultats obtenus avec 400 chromosomes (le cas idéal de la section 3.2) et 600 chromosomes n'est pas significative au point d'accepter 200 chromosomes supplémentaires dans ces calculs.

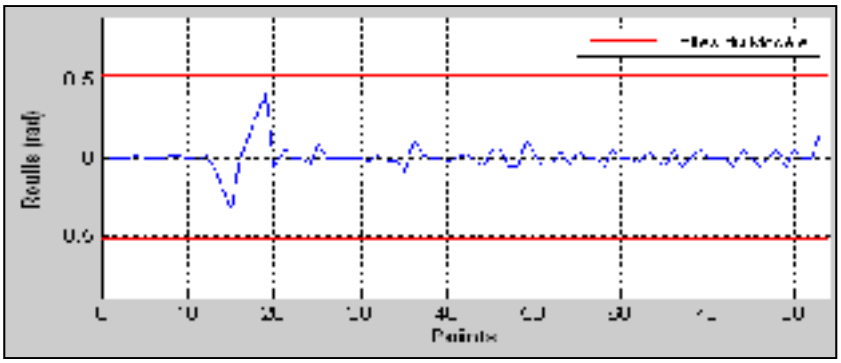

Figure 3.18 Roulis versus le point de trajectoire pour 600 chromosomes.

3.4 Analyse du nombre d'itérations (5 et 25) pour chaque sous-période

Il s'agit ici de confirmer le choix de 10 pour le nombre d'itérations de chaque sous-période. Pour cette confirmation, nous avons procédé à une série de tests dont nous allons montrer les limites extrêmes. Les conditions ou spécifications regroupent :

- **un nombre d'itérations des sous-périodes de 5 ou 25;**
- un taux de mutation de 0.001;
- un nombre de chromosomes de 400;
- un taux d'approchement des points de rendez-vous de 85%;
- des conditions initiales n'avantageant aucune des données du tableau 3.1.

Les figures 3.19 à 3.21 nous montrent les résultats obtenus pour 5 itérations. Il s'agit de l'évolution des paramètres spatiaux (latitude, longitude et altitude) pour 5 itérations par sous-périodes. Ce qui est le plus flagrant dans cette configuration c'est ce qui se passe entre les points de rendez-vous 3 et 5. On n'a pas ce genre d'évolution dans le cas idéal. Les données sont capables de se rapprocher plus de leur courbes d'origine générant un mouvement de va-et-vient pour la trajectoire. Pour la longitude par exemple, l'algorithme devrait obliger les données à se déplacer en ligne droite et pas à revenir vers la courbe pour repartir. Ce genre de disparité (entre les points de rendez-vous 3 et 5 sur la figure 3.20) et de va-et-vient aussi prononcé, montre une instabilité dans l'algorithme et dans sa recherche de solution.

En ce qui concerne l'altitude, l'algorithme réagit de la même manière que pour le cas idéal. Ce qui est le plus important à remarquer, c'est que dans ce cas, le principe des algorithmes génétiques n'a plus le temps d'opérer et l'initialisation est la partie qui fait le plus évoluer la trajectoire. On pourrait presque analyser l'évolution de notre trajectoire comme une succession d'initialisations, ce qui donne des résultats incertains et rend les calculs extrêmement longs.

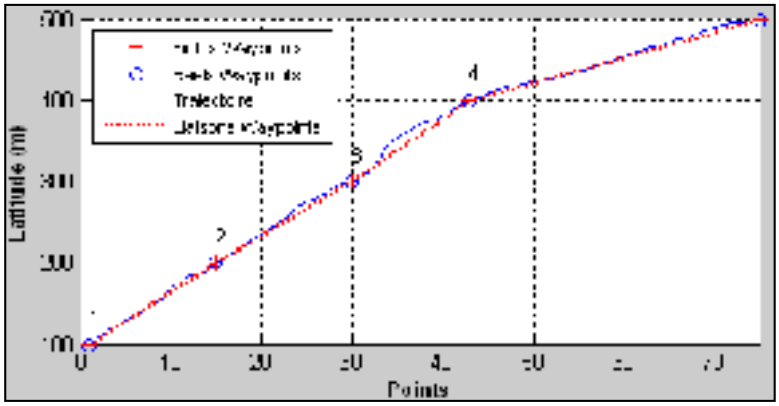

Figure 3.19 Latitude versus les points de trajectoires pour 5 itérations.

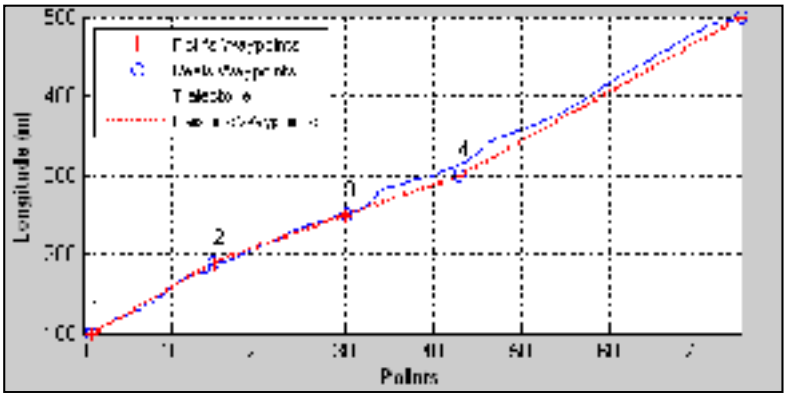

Figure 3.20 Longitude versus les points de trajectoires pour 5 itérations.

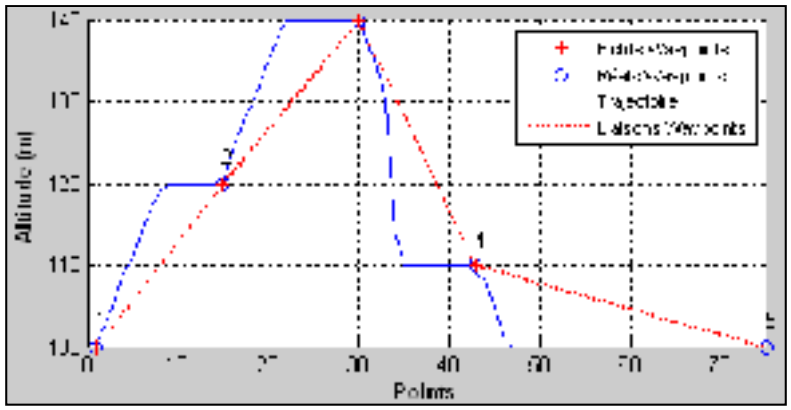

Figure 3.21 Altitude versus les points de trajectoires pour 5 itérations.

Les figures 3.22 à 3.24 nous montrent les résultats pour des paramètres spatiaux (latitude, longitude et altitude) pour 25 itérations par sous-périodes. Les résultats sont assez poches des résultats obtenus dans le cas idéal présenté dans la section 3.2. Pourtant certaines différences sont notables. Les figures 3.22 et 3.23 nous montrent des trajectoires s'éloignant de la courbe entre les points de rendez-vous 1 et 2. Ces changements moins radicaux dans le cas idéal sont dus aux contraintes d'angles d'approche qui sont alors prises en compte bien plus tôt qu'il ne le faudrait. On perd ici l'efficacité des sous-périodes car elles sont trop grandes et la dispersion des points et des solutions n'en est que plus mauvaise comme expliqué dans la section 2.8.

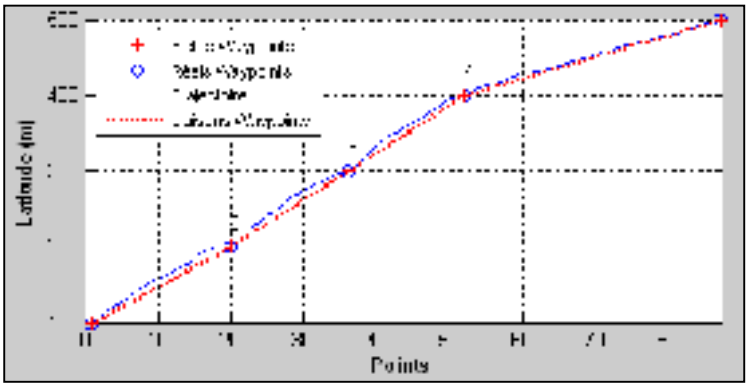

Figure 3.22 Latitude versus le point de trajectoire pour 25 itérations.

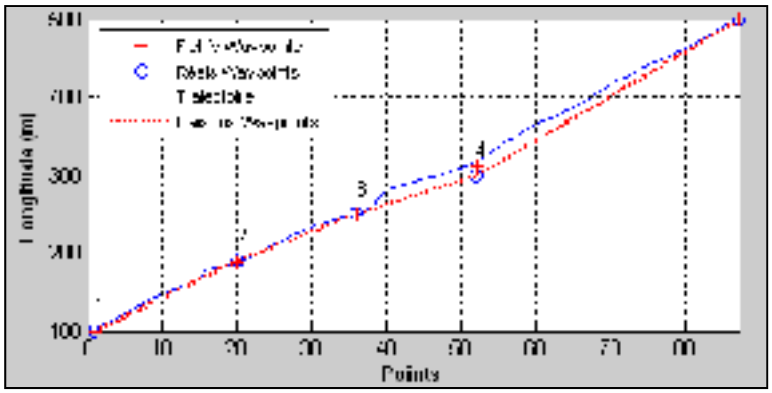

Figure 3.23 Longitude versus le point de trajectoire pour 25 itérations.

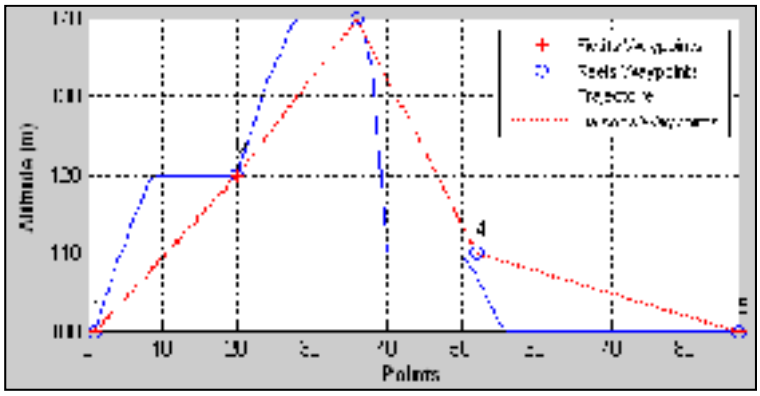

Figure 3.24 Altitude versus le point de trajectoire pour 25 itérations.

Nous avons donc choisi 10 itérations.

3.5 Analyse pour le taux de mutation de 0.1

Il s'agit ici de montrer le choix de la mutation, et pour illustrer cette influence, nous avons donc choisi un taux plus élevé de mutation de 0.001 pour analyser son effet. Les conditions regroupent :

- un nombre d'itérations des sous-périodes de 10;

- **un taux de mutation de 0.1;**
- un nombre de chromosomes de 400;
- un taux d'approchement des points de rendez-vous de 85%;
- des conditions initiales n'avantageant aucune des données du tableau 3.1.

Les figures 3.25 à 3.30 nous montrent les effets du taux de mutation plus élevé sur les six dimensions de l'avion. L'intérêt de la mutation est d'augmenter le brassage génétique et l'exploration de l'espace des solutions. Il y a plusieurs facteurs à considérer dans le cas d'un taux de mutation élevé, tel montré dans ces figures.

Un de ces facteurs est l'apparition de solutions extrêmes. Ces solutions sont causées par une mutation et peuvent passer au travers des contraintes fixées par la fonction d'évaluation de notre algorithme. Dans les figures 3.25 et 3.26, on remarque un éloignement soudain de la route rectiligne entre les points 4 et 5. Un point de trajectoire a muté et se retrouve bien plus loin de son précédent. Ce genre de solution est acceptée car elle ne viole aucune contrainte, pourtant ce n'est pas une solution recherchée. Elle évolue trop vite et aléatoirement. Dans notre cas, lorsque l'on désire explorer l'espace des solutions, notre but n'est pas d'atteindre le plus rapidement possible le point de rendez-vous suivant, mais de l'atteindre correctement.

Le genre de solutions que cette mutation peut entraîner, est aussi l'apparition des angles limites. Certes les angles limites resteront sous l'angle maximal des 30 degrés, mais il y aura plus d'angles de grandes valeurs, ce qui n'est pas recommandé. La figure 3.28 des variations de lacet nous montre des angles plus grands que ceux calculés dans le cas idéal. Seul les angles de tangage et les altitudes semblent avoir été épargnés (figures 3.27 et 3.30). La contrainte d'ajustement d'altitude demandant à l'algorithme de l'atteindre rapidement la mutation ne peut accélérer plus le processus.

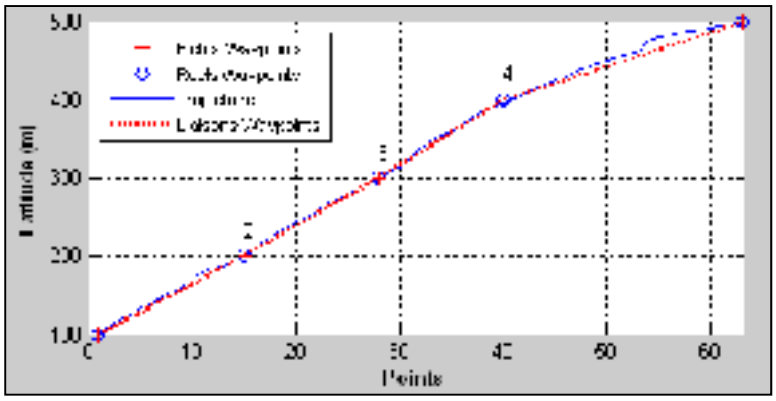

Figure 3.25 Latitude versus le point de trajectoire pour une mutation de 0.1.

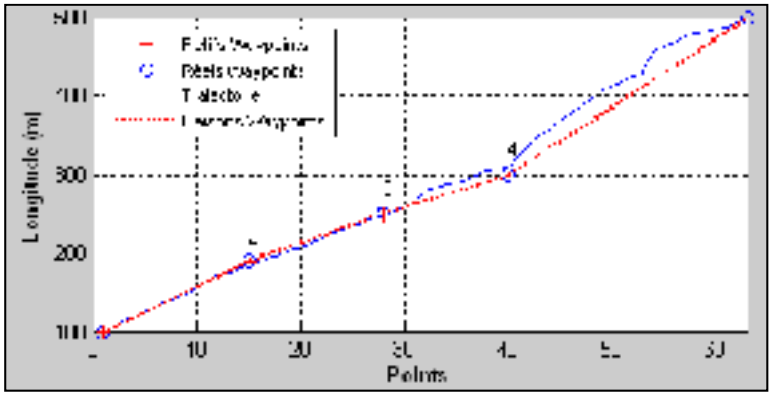

Figure 3.26 Longitude versus le point de trajectoire pour une mutation de 0.1.

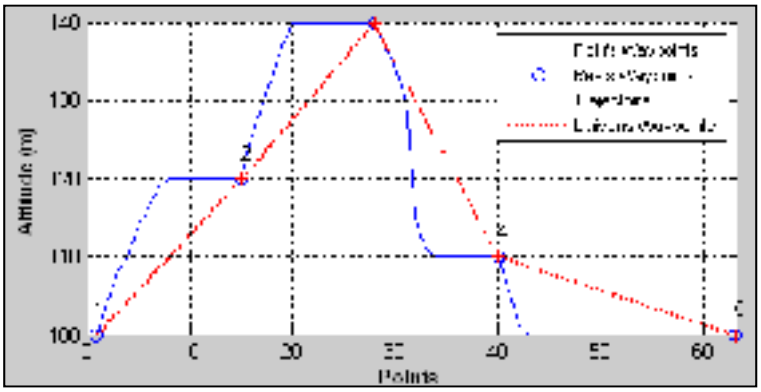

Figure 3.27 Altitude versus le point de trajectoire pour une mutation de 0.1.

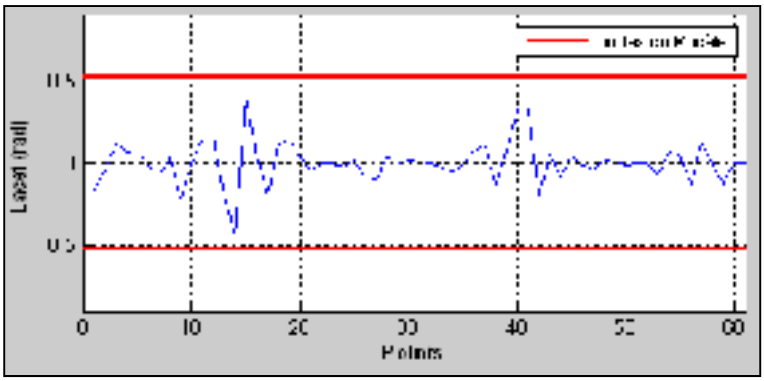

Figure 3.28 Lacet versus le point de trajectoire pour une mutation de 0.1.

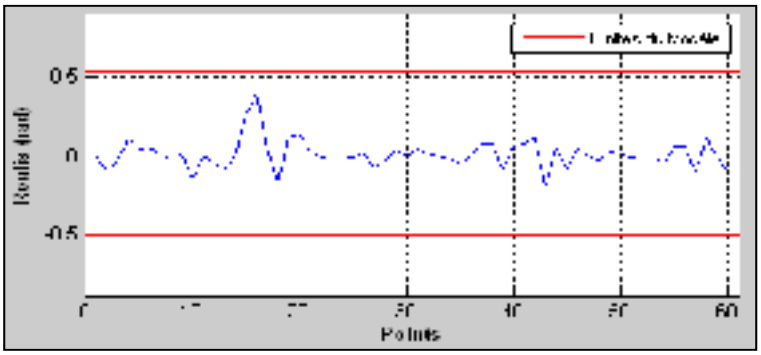

Figure 3.29 Roulis versus le point de trajectoire pour une mutation de 0.1.

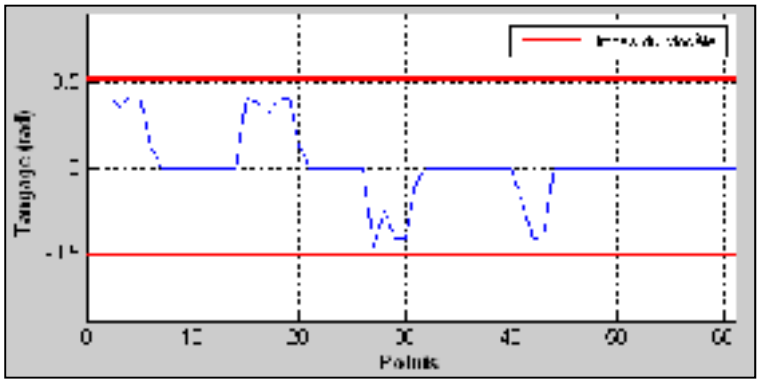

Figure 3.30 Tangage versus le point de trajectoire pour une mutation de 0.1.

3.6 Analyse pour le rapprochement des points de rendez-vous (50% et 100%)

Nous allons illustrer quelques exemples de routes pour analyser les effets du rapprochement des points de rendez-vous sur les solutions calculées. On obtient ici les résultats de la méthodologie énoncés dans la section 1.4. Les spécifications regroupent :

- un nombre d'itérations des sous-périodes de 10;
- un taux de mutation de 0.001;
- un nombre de chromosomes de 400;
- **des taux d'approchement des points de rendez-vous de 50% et 100% (choix pris pour montrer des exemples de la méthodologie expliquée dans la section 1.4);**
- des conditions initiales n'avantageant aucune des données du tableau 3.1.

Le principe était de montrer la différence entre la route que l'on demande et la route calculée, en démontrant la limite acceptable entre ces routes. Il est hors de question de dériver trop loin de la route initiale. Les résultats sont affichés dans les figures 3.31 à 3.34. Ce qui est important de voir sur la figure 3.33, est le fait que la trajectoire entre les points de rendez-vous 3 et 5 est presque une ligne dans le cas d'un taux de proximité de 50%. Le point 4 est pratiquement effacé de la trajectoire. Ceci n'est pas une trajectoire admissible car on doit passer à la proximité de chaque point. D'un autre côté, pour un taux de proximité de 100%, la figure 3.31 nous montre une trajectoire qui colle à celle voulue, ce qui prouve que la proximité des deux trajectoires ne dépend que du taux de proximité, et non de l'algorithme de calcul. Il ne tient donc qu'à déterminer la bonne proximité.

Les figures 3.32 et 3.34 nous montrent les variations demandées sur la longitude (la latitude est similaire à la longitude et la l'altitude restent inchangée), en tenant compte de notre méthodologie. L'algorithme se comporte de la même façon, mais il suit le point de rendez-vous recalculé. Le taux de 85% (montrée dans le cas idéal) fut un compromis pour rester proche du point tout en économisant le trajet. L'introduction de la vitesse et de la consommation dans le projet, permettra d'affiner ce taux et de se servir pleinement de cette

option. Ce sont des contraintes supplémentaires que l'on pourrait ajouter dans le futur du projet.

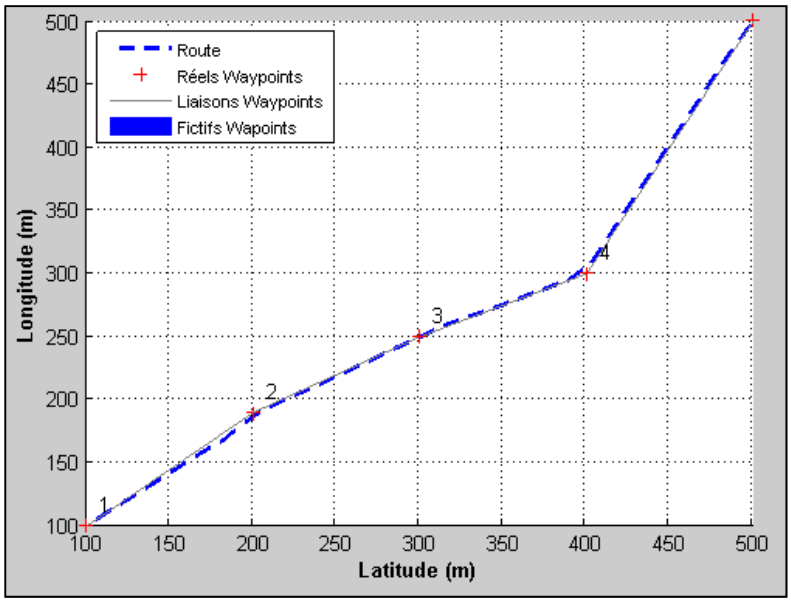

Figure 3.31 Route calculée avec une proximité de 100% (longitude versus latitude).

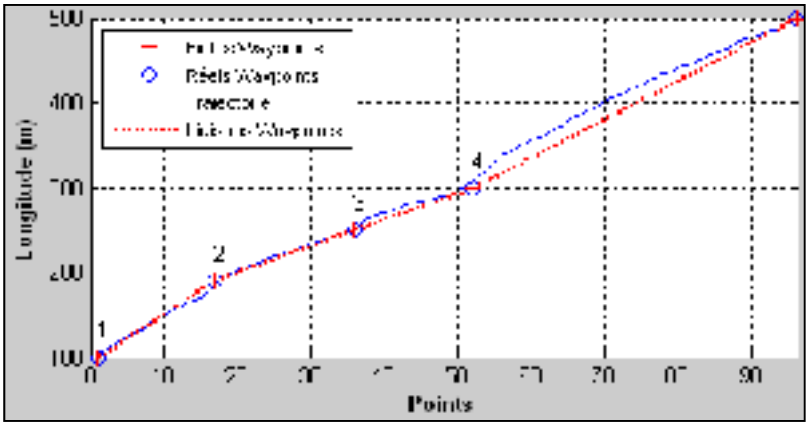

Figure 3.32 Longitude versus le point de trajectoire calculée avec une proximité de 100%.

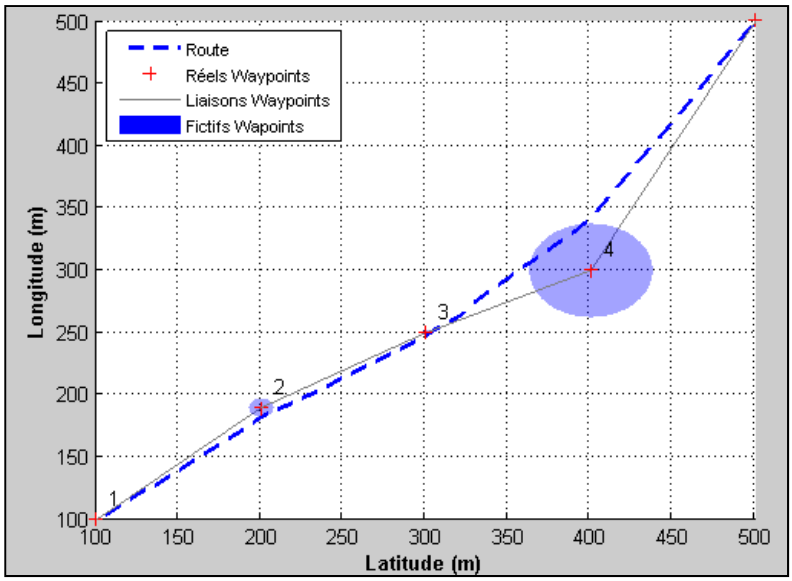

Figure 3.33 Route calculée avec une proximité de 50% (longitude versus latitude).

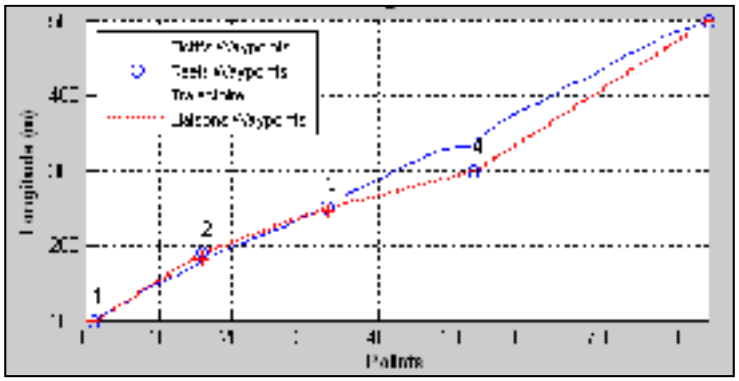

Figure 3.34 Longitude versus le point de trajectoire calculée avec une proximité de 50%.

3.7 Avantager des paramètres spatiaux lors de l'initialisation de la route et de l'altitude

Nous allons exposer quelques exemples où certains des trois paramètres spatiaux seront avantagés, lors de l'initialisation des algorithmes génétiques pour trouver leurs effets sur la trajectoire. Ces conditions regroupent :

- un nombre d'itérations des sous-périodes de 10;
- un taux de mutation de 0.001;
- un nombre de chromosomes de 400;
- un taux d'approchement des points de rendez-vous de 85%;
- **une initialisation des données au commencement des algorithmes génétiques en avantageant la route (longitude et latitude) ou l'altitude.**

Le but de cette section est de montrer la variation de l'initialisation de l'algorithme génétique. Nous allons d'abord avantager la route et ensuite l'altitude, pour déterminer ensuite l'influence de l'initialisation sur les performances de l'algorithme. L'avantage lors de l'initialisation est de l'ordre de 300%. Par exemple si on a une donnée de départ de 123 mètres et que la troncature s'exécute à 120 mètres, une initialisation normale pourrait donner 126 mètres (selon les principes énoncés dans la section 2.2). Ici, en utilisant la donnée avantagée, on obtiendra 138 mètres. L'algorithme donne dès le départ, une solution plus proche de la solution finale. Dans un premier temps, les figures 3.35 à 3.40 nous montrent les effets d'une telle manœuvre sur la latitude et la longitude. Cet effet se fait ressentir sur les six dimensions de notre modèle.

La première chose que l'on remarque c'est le nombre de points dont l'algorithme a besoin pour résoudre le problème. Dans le cas optimal environ 80 points sont utilisés. Ici il ne faut que 35 points à l'algorithme pour parvenir au dernier point de rendez-vous de la trajectoire. La solution est atteinte plus rapidement par l'algorithme car les initialisations l'amènent plus vite sur le point de rendez-vous suivant. Ceci peut paraître intéressant si l'on augmente la taille des données. Mais il s'agit avant tout d'une perte de précision.

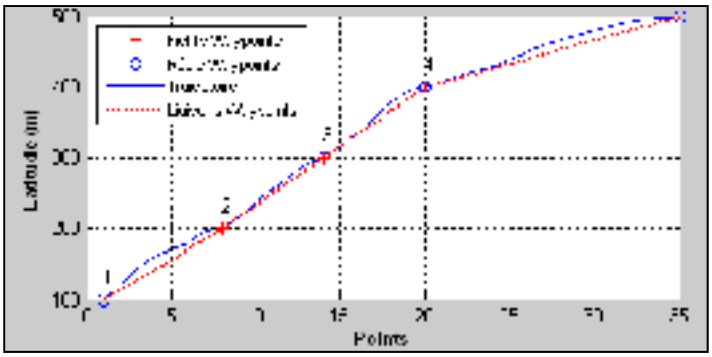

Figure 3.35 Latitude versus le point de trajectoire calculée en avantageant l'initialisation de la route.

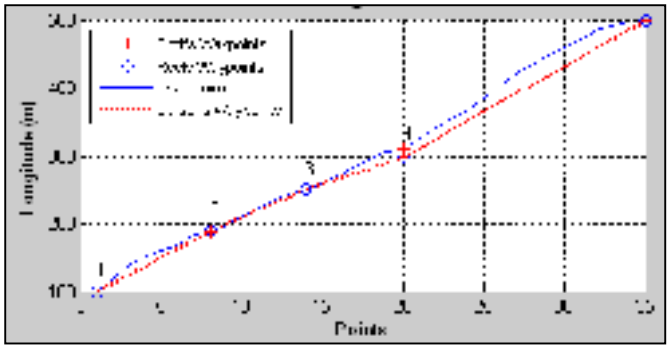

Figure 3.36 Longitude versus le point de trajectoire calculée en avantageant l'initialisation de la route.

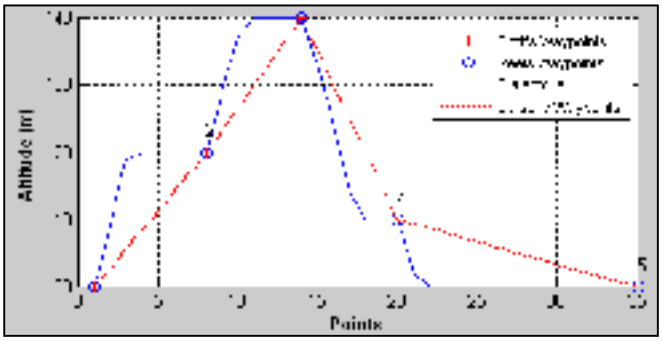

Figure 3.37 Altitude versus le point de trajectoire calculée en avantageant l'initialisation de la route.

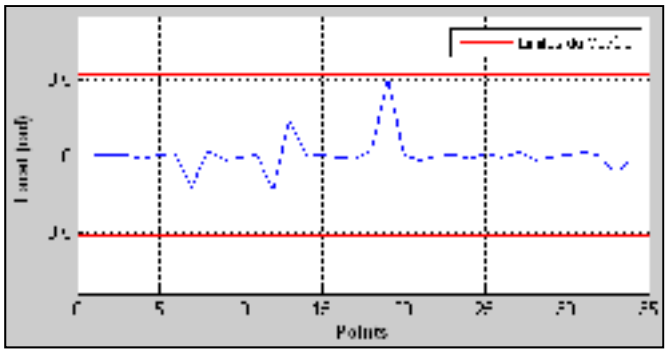

Figure 3.38 Lacet versus le point de trajectoire calculé en avantageant l'initialisation de la route.

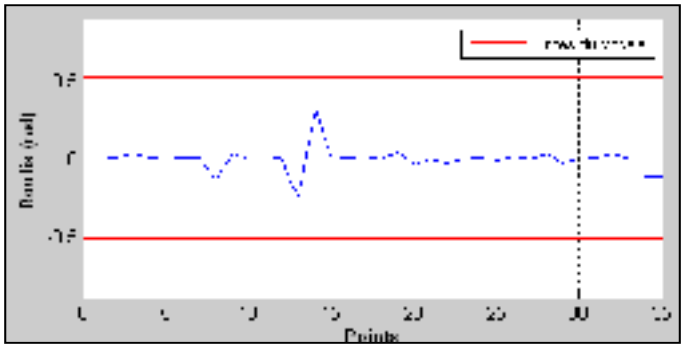

Figure 3.39 Roulis versus le point de trajectoire calculé en avantageant l'initialisation de la route.

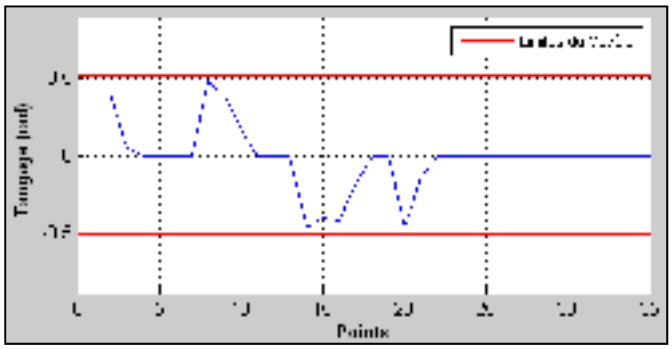

Figure 3.40 Tangage versus le point de trajectoire calculé en avantageant l'initialisation de la route.

Les figures 3.41 à 3.46 nous montrent les résultats pour une initialisation avantagée de l'altitude. On remarque que les données de route (latitude, longitude, lacet, et roulis) semblent inchangées. Ce qui donne les plus grands changements sont les données d'altitude. L'altitude varie de façon chaotique et non contrôlée. Il semble que l'augmentation de l'initialisation de l'altitude fausse son évolution.

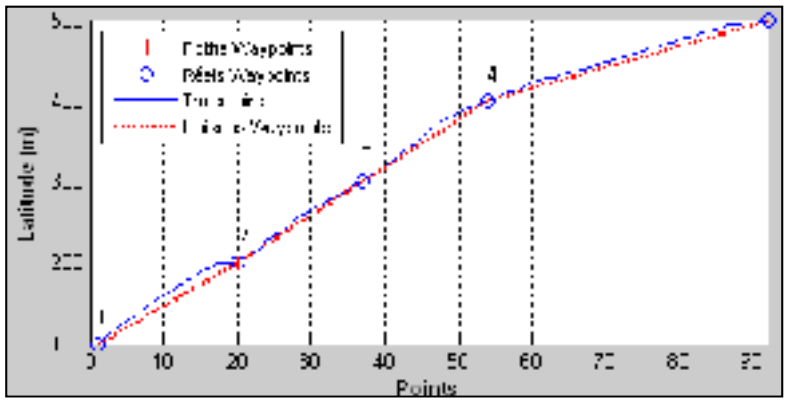

Figure 3.41 Latitude versus le point de trajectoire calculée en avantageant l'initialisation de l'altitude.

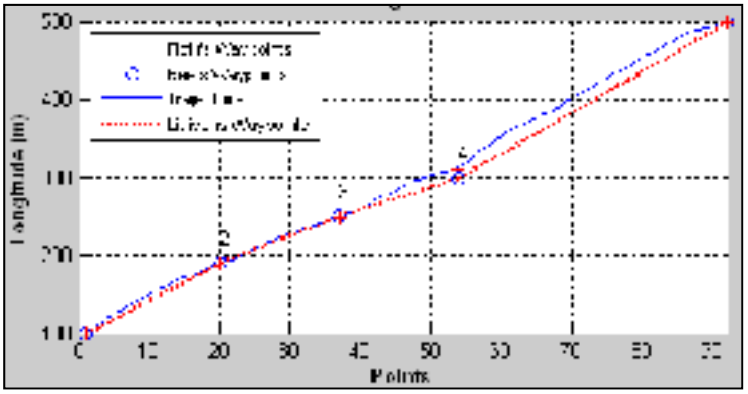

Figure 3.42 Longitude versus le point de trajectoire calculée en avantageant l'initialisation de l'altitude.

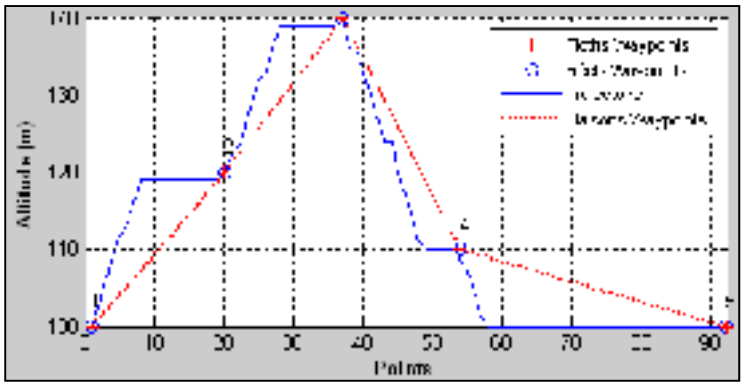

Figure 3.43 Altitude versus le point de trajectoire calculée en avantageant l'initialisation de l'altitude.

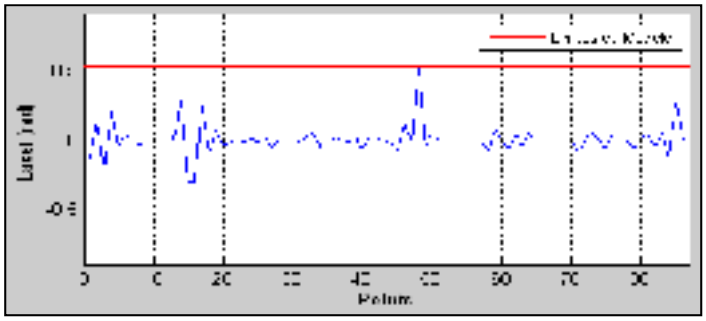

Figure 3.44 Lacet versus le point de trajectoire calculée en avantageant l'initialisation de l'altitude.

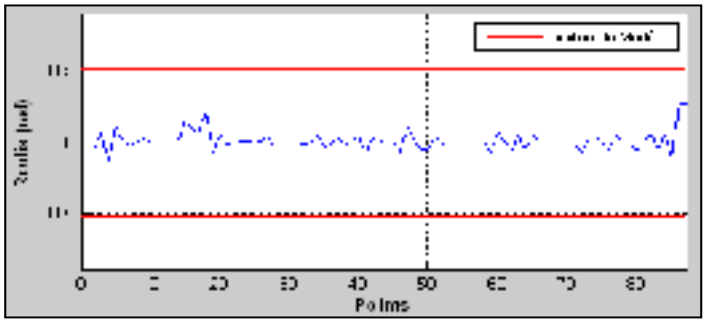

Figure 3.45 Roulis calculée versus le point de trajectoire en avantageant l'initialisation de l'altitude.

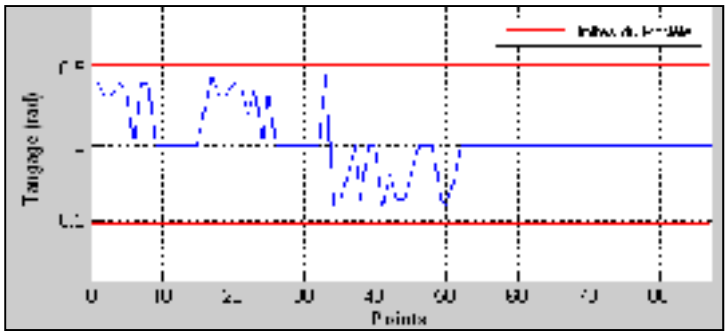

Figure 3.46 Tangage versus le point de trajectoire calculée en avantageant l'initialisation de l'altitude.

CONCLUSION

Une première étape dans les calculs de trajectoires en trois dimensions pour un avion modélisé en six dimensions à donc été réalisée. Les algorithmes génétiques furent la réponse aux nombreuses contraintes de nos trajectoires. Ainsi les trajectoires calculées ont été conditionnées sur les angles de lacet, roulis et tangage, sur l'ajustement d'altitude, le non dépassement de trajectoire, la stabilité de route, l'atteinte des points de rendez-vous et autres paramètres. Tous ces paramètres étaient contrôlés séparément, ce qui nous amène au point important de cette étude, qui est la capacité des algorithmes génétiques à générer des solutions suivant un grand nombre de contraintes. Bien que nous n'en n'ayons exploré que quelques unes, il reste facile d'en rajouter. Au final, notre algorithme est en mesure de lier des points de rendez-vous intelligemment et de fournir les données nécessaires au pilote.

Maintenant que l'algorithme est en place, les hypothèses et limites de ce projet peuvent être dépassées dans le futur. La première étude qu'il serait intéressant de faire concerne la modélisation structurelle et aérodynamique de l'avion. En effet, nos hypothèses se limitent à confondre l'avion avec un point. Une étude plus approfondie sur ses limitations structurelles et ses comportements en vol nous donnerait un profil aérodynamique de comportement, qui pourrait alors directement être utilisé pour contraindre les algorithmes génétiques. La deuxième étude concerne l'augmentation des dimensions dans le temps. Il serait alors possible de varier la vitesse et la consommation de l'avion, et obtenir un véritable calculateur de trajectoire.

La finalisation de ce projet étant directement liée à l'augmentation de la précision de position et de vol de l'avion, elle est essentielle pour l'avenir. La consommation de carburant et les horaires de vol sont directement liés aux calculateurs de trajectoire. De même les nouvelles idées comme l'approche en descente continue pour les atterrissages demandent l'utilisation de calculateurs plus performants. La poursuite de cette voie permet donc de débloquer de nouvelles approches de vols plus économiques et bénéfiques du point de vue écologique.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Cordenonsi A. Z., Couto Barone D. A. et Resende Thielo M. 1997. *Trajectories Control of a Projectile Using Genetic Algorithms in one Simulated Environment*, Universidade Federal do Rio Grande do Sul Instituto de Informática Curso de Pós-Graduação em Ciência da Computação, pp. 1289-1292.

[2] Durand N. et Gotteland J.-B. 2003. *Algorithmes Génétiques Appliqués à la Gestion du Trafic Aérien*, Laboratoire d'Optimisation Globale Centre d'études de la navigation aérienne / École Nationale de l'Aviation Civile, J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes, Vol. 2, Hors-Série 1.

[3] Spellucci, 1998, *An SQP Method for General Nonlinear Programs Using only Equality Constrained Subproblems*, vol. 82, n°3, pp. 413-448

[4] Holland J. H. 1992. *Adaptation in Natural and Artificial Systems*, MIT Press, pp. 1-211

[5] Jorris T. R. September 2007. *Common Aero Vehicle Autonomous Reentry Trajectory Optimization Satisfying Waypoint and No-Fly Zone Constraint*, Major, USAF AFIT/DS/ENY/07-04 Department of the Air Force Institute of Technology Wright-Patterson Air Force Base, Ohio.

[6] Narendra K. S. et Parthasarathy K. March 1990. *Identification and control of dynamical systems using neural networks,* IEEE Transactions on Neural Network, Vol. 1(1), pp. 4-27.

[7] Spitzer C. R. 2000. *The Avionics Handbook*, Chapter 15.

[8] Sugeno M. 1985. *Industrial Application of Fuzzy Control*, New York: Elsevier Science Pub.

[9] Vormer F. J, Mulder M., Van Paassen M. M., et Mulder J. A., 2006. *Optimization of Flexible Approach Trajectories Using a Genetic Algorithm*, Journal of Aircraft, Vol. 43(4), pp. 941-952.

[10] Yokoyama N., et Suzuki S., 2005. *Modified Genetic Algorithm for Constrained Trajectory Optimisation,* Journal of Guidance, Control, and Dynamics, Vol. 28(1), University of Tokyo, Tokyo 113-8656, Japan.

[11] Yokoyama N. et Susuki S., 2001. *Flight Trajectory Optimization Using Genetic Algorithm Combined with Gradient Method*, Information Technology for Economics & Management ITEM, Vol. 1(1), Paper 10, pp. 1-8.

[12] Zadeh L. April 1988. *Fuzzy Logic. IEEE Computer*, Vol. 21, pp. 83-92.