Optimisation numérique de profils d'aile par algorithmes génétiques et jeux de Nash

Hélène Lièvre, Jean-Antoine Desideri, Abderrahmane Habbal

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Optimisation numérique de profils d'aile par Algorithmes Génétiques et Jeux de Nash.

Hélène Lièvre - Jean-Antoine Désidéri - Abderrahmane Habbal

N° 4275

septembre 2001

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--4275--FR

INRIA
SOPHIA ANTIPOLIS

Optimisation numérique de profils d'aile par Algorithmes Génétiques et Jeux de Nash.

Hélène Lièvre - Jean-Antoine Désidéri - Abderrahmane Habbal

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Sinus

Rapport de recherche n° 4275 — septembre 2001 — ?? pages

Résumé : Ce rapport présente une série d'expériences numériques visant à résoudre un problème d'optimisation de formes en aérodynamique. Il s'agit de déterminer le profil présentant le meilleur compromis entre un profil adapté au régime de croisière (profil à faible traînée en transonique à faible incidence) et un profil adapté au décollage (profil à grande portance en subsonique et grande incidence). Pour cela on utilise la théorie de Nash: chaque joueur travaille sur un des cas de figure en utilisant un algorithme génétique. Au bout de quelques générations les joueurs s'échangent une partie de leurs résultats jusqu'à ce qu'ils aboutissent à un profil qu'ils ne peuvent plus, ni l'un ni l'autre, améliorer: c'est l'équilibre de Nash. Le profil ainsi obtenu est un bon compromis entre les 2 solutions mono-critères.

Mots-clés : algorithme génétique, théorie de Nash, profil d'aile, optimisation multi-critères

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Numerical optimization of shape design by Genetic Algorithms and Nash games.

Abstract: This report presents a series of numerical experiments in which a problem of shape optimization in field aerodynamic is solved.We are looking for an airfoil which realizes the best compromise between a cruising speed adapted airfoil (with low drag in transonic flow and low incidence) and a takeoff adapted airfoil (with high lift in subsonic flow and high incidence). We use therefore the Nash theory: each player operates on one of the cases by using a genetic algorithm. After some generations, the players exchange part of their results until they lead to a profile that they cannot improve any more: it is the Nash equilibrium. The profil thus obtained is a good compromise between the two single-criterion solutions.

Key-words: genetic algorithm, Nash theory, airfoil shape, multi-criterion optimizatio

Table des matières

1 Introduction

Le but de cette étude est la maîtrise d'outils d'optimisation en vue de leur application en aérodynamique, et plus particulièrement, la détermination d'un profil d'aile présentant le meilleur compromis entre un profil optimal pour le décollage (régime subsonique) et un profil optimal pour le régime de croisière d'un avion commercial (régime transsonique). Il s'agit donc d'un problème multi objectif.

Les problèmes multi objectifs sont courants: la plupart du temps les concepteurs cherchent à optimiser plusieurs paramètres concurrents comme par exemple maximiser des performances tout en minimisant les coûts, l'énergie dépensée ou en maximisant les rendements. Plusieurs critères sont donc souvent à prendre en compte simultanément. Pour ces cas-là, les méthodes d'optimisation mono objectif ne sont plus suffisantes. Les méthodes classiques comme la méthode des poids [1], sont limitées en robustesse et efficacité lorsque les intervalles de recherche des candidats sont vastes. Il a été montré [2] que les algorithmes génétiques, utilisant des populations de solutions, constituent des méthodes très robustes pour résoudre les problèmes dont les solutions peuvent se situer dans de grands intervalles. Plus récemment, les algorithmes génétiques ont été implémentés et combinés avec la théorie des jeux en optimisation multi-objectifs et ont conduit à de nombreuses applications [3] [4] [5].

Nous avons tout d'abord expérimenté un algorithme génétique ; les algorithmes génétiques sont des algorithmes d'optimisation qui proposent une ou plusieurs solutions à un problème donné. Leur principe s'inspire de la sélection naturelle, c'est-à-dire que les solutions les mieux adaptées au milieu, celles répondant le mieux aux objectifs poursuivis, se croisent et donnent naissance à des solutions encore mieux adaptées. Nous allons voir que ces algorithmes peuvent être utilisés par exemple pour déterminer le profil de moindre traînée en régime de croisière ou celui de plus grande portance en régime correspondant à un décollage.

D'autre part, nous nous sommes intéressés à l'équilibre de Nash : cette technique d'optimisation provient de la théorie des jeux et consiste à traiter les différents objectifs de manière concurrentielle comme des joueurs indépendants échangeant de l'information partielle. La recherche de l'équilibre de Nash se fait par échange des résultats obtenus pour chaque fonction objectif travaillant

avec une partie seulement des variables, les autres étant fixées par les résultats obtenus pour les autres fonctions objectifs. Cet équilibre est atteint quand l'optimisation de chaque fonction conduit toujours au même individu. Nous avons utilisé cette méthode pour déterminer un profil d'aile qui réalise un bon compromis entre les 2 profils d'ailes trouvés avec l'algorithme génétique. Dans cette approche, le profil candidat est paramétrisé par des courbes de Bézier, dont les points de contrôle représentent les chromosomes pour les individus de l'algorithme génétique. On fait une répartition avant-arrière des points de contrôle entre les 2 joueurs de Nash [6] qui sont associés aux 2 objectifs ; l'un des joueurs optimise le premier critère en modifiant l'avant du profil en gardant l'arrière trouvé par le deuxième joueur et ce deuxième joueur optimise le deuxième critère en modifiant l'arrière du profil en gardant l'avant trouvé par le premier joueur.

Ces résultats sont ensuite comparés avec ceux obtenus dans [5] par une approche de reconstruction de pression (problème inverse).

2 Généralités sur les outils d'optimisation utilisés

2.1 Les algorithmes génétiques

Les algorithmes génétiques (AG) s'inspirent à la fois des mécanismes de la sélection naturelle et de la génétique. On identifie le problème à un environnement donné et les solutions à des individus évoluant dans cet environnement. A chaque génération, on ne retient que les individus les mieux adaptés à l'environnement. Au bout d'un certain nombre de générations, les individus restants sont particulièrement adaptés à l'environnement donné. L'adaptation de l'individu à son milieu est quantifié par l'évaluation d'une fonction appelée fonction objectif ou fonction d'adaptation. C'est la fonction à optimiser (à minimiser ou à maximiser). Un algorithme génétique peut proposer des solutions pour des problèmes à un ou plusieurs objectifs. Les individus sont les éléments à optimiser (pour nous ce sont les profils d'ailes), chaque individu est caractérisé par des variables qui peuvent être codées en binaire ou en réel (pour nous ce sont les ordonnées des points de contrôle définissant le profil) (voir aussi [2], [7], [8]).

Le principe de l'algorithme est simple: une première population d'individus est générée aléatoirement. L'individu est caractérisé par son chromosome qui est le codage de toutes les variables le caractérisant. A chaque génération, on sélectionne les meilleurs individus selon la (ou les) fonction(s) objectif(s) et certains de ces individus se reproduisent ou mutent, puis on itère le processus.

- l'opérateur de sélection : il existe plusieurs méthodes pour sélectionner les individus les mieux adaptés. La plus simple consiste à prendre 2 individus au hasard dans la population, à comparer leur adaptation et à sélectionner le mieux adapté. C'est la méthode dite de tournoi. La méthode la plus connue est nommée "roulette-wheel", c'est une sélection proportionnelle au niveau de l'adaptation de l'individu: on calcule l'adaptation relative qui est l'adaptation de l'individu sur la somme des adaptations. On affecte ensuite à chaque individu une partie distincte du segment [0,1] et de longueur égale à son adaptation relative. On génère ensuite aléatoirement un réel entre 0 et 1, on sélectionne ensuite l'individu dont le segment contient ce chiffre. De nombreuses méthodes plus complexes existent par ailleurs, [2], [7], [8].

- l'opérateur de croisement : en binaire la méthode la plus courante consiste à prendre un nombre aléatoire k compris entre 1 et la longueur l de la chaîne. A partir des chaînes de caractères de deux individus parents, on crée deux nouvelles chaînes par permutation de tous les caractères compris entre les positions k+1 et l des chaînes. On peut aussi choisir plusieurs lieux de coupe et échanger les parties de chaîne comprises entre les coupes (croisement multiple). Si l'on travaille en codage réel, le croisement le plus courant se fait par échange de gènes ; on peut échanger des gènes au hasard ou générer de nouveaux gènes par combinaison linéaire des gènes des 2 parents (croisement arithmétique).

- l'opérateur de mutation : il vise à augmenter la diversité de la population pour éviter les optimums locaux, en modifiant de façon aléatoire et occasionnelle une partie de la population. Le principe est de choisir une valeur de remplacement aléatoire pour l'un des gènes des individus de la population concernée. Lors d'un codage binaire des chaînes, cela revient à changer un 1 en 0 et vice versa. Ceci permet à la fois de pouvoir explorer le domaine complet de variation des variables et également d'éviter un appauvrissement de la population qui pourrait entraîner, en particulier, la convergence vers un extremum local. Il existe d'autres méthodes de mutation comme la mutation variable, qui diminue l'intervalle de génération des nouveaux individus lorsque l'on avance dans les générations.

Pour converger plus régulièrement et plus rapidement, de nombreuses procédures peuvent être mises en place. En voici quelques-unes :

- procédure élitiste : on travaille de façon à ce que la fonctionnelle coût ne puisse que décroître. Si à une génération i, on a trouvé un meilleur chromosome on le stocke pour les générations suivantes jusqu'à ce que l'on en trouve un meilleur.

- procédure de mise à l'échelle : lorsque l'adaptation maximale se trouve proche de son adaptation moyenne, l'algorithme génétique se révèle peu efficace car tous les individus ont alors quasiment les mêmes chances d'être sélectionnés. On se propose alors d'étaler l'échelle des adaptations après avoir recalé le minimum à zéro. On multiplie ensuite les adaptations des individus par un coefficient qui diminue lorsque l'on avance dans les générations de sorte que l'on valorise de plus en plus les meilleures adaptations.

- procédure «niche» ou sharing : cette procédure permet d'éviter de converger vers un seul extremum, c'est-à-dire qu'elle permet de laisser une chance à

d'autres bonnes solutions. Pour cela on favorise les solutions qui ont un faible nombre de proches voisins. La proximité des solutions est évaluée grâce à la distance d de Hamming (nombre de bits différents entre deux individus). Lorsque cette distance est inférieure à un nombre donné, on implémente un nombre m_i (toujours inférieur à 1) et la fonction objectif est divisée par ce nombre donc elle se trouve augmentée . Ainsi un individu très entouré de proches solutions devient moins bien adapté au milieu (cas de la minimisation de la fonction objectif).

$$m_i = \sum_j S(d_{ij}) \quad avec \quad S(d) = \begin{cases} 1 - (\dfrac{d}{\sigma_{share}}) & si \quad d < \sigma_{share} \\ 0 \quad si & d > \sigma_{share} \end{cases}$$

- procédure de pénalisation: dans certains problèmes toutes les solutions de l'espace ne sont pas réalisables. Par exemple une solution de profil d'aile d'avion présentant une intersection de l'extrados et de l'intrados n'est pas acceptable. Il faut alors pénaliser ces solutions afin qu'elles ne soient pas sélectionnées pour le croisement. Il existe différentes manières de résoudre de tels problèmes; sans rentrer dans le détail, un individu ne respectant pas les contraintes sera considéré comme moins bien adapté, voire il sera éliminé (procédure «kill»).

2.2 L'équilibre de Nash

La théorie des jeux concerne l'étude des situations dans lesquelles les individus interagissent dans un environnement d'interdépendance stratégique. Lorsqu'aucune stratégie ne domine l'autre, on a l'habitude de recourir à l'équilibre de Nash.

Du nom de son auteur John Nash, lauréat du prix Nobel de sciences économiques (1994), l'équilibre de Nash se compose d'un certain nombre de stratégies telles qu'aucun joueur ne peut faire mieux en modifiant unilatéralement la sienne. Ainsi pour l'optimisation de G objectifs, la stratégie de Nash utilise G joueurs, chacun optimisant son propre critère. De plus chaque joueur doit optimiser son critère sachant que tous les autres critères sont fixés par le reste des joueurs. Lorsqu'aucun joueur ne peut plus améliorer son critère, cela signifie que le système a atteint un état d'équilibre appelé Equilibre de Nash. Avec une approche classique, l'équilibre de Nash n'est pas facile à trouver et devient pratiquement impossible à déterminer lorsque les critères sont

des fonctions non-différentiables. Par contre, l'utilisation de Nash couplé avec les algorithmes génétiques présente un intérêt certain.

L'optimisation s'effectue de la manière suivante :

Appelons s=XY la chaîne représentant la solution potentielle pour une optimisation bi-objectif, où X correspond au premier critère et Y au second. La première idée est d'assigner l'optimisation de X au joueur 1 et celle d'Y au joueur 2. Alors, d'après la théorie de Nash, le joueur 1 optimise (à l'aide d'un algorithme génétique, par exemple) s en modifiant X avec respect du premier critère, tandis que Y est fixé par le joueur 2. Et inversement, le joueur 1 optimise s en modifiant Y tandis que X est fixé par le joueur 1.

Le pas suivant consiste à créer 2 populations différentes, une pour chaque joueur, avec lesquelles ils vont optimiser leur critère. Appelons X_{k-1}et Y_{k-1} les meilleures valeurs trouvées respectivement par les joueurs 1 et 2 à la génération $k-1$. Alors le joueur 1 optimise X_k en utilisant Y_{k-1} pour évaluer s, et le joueur 2 optimise Y_k en utilisant X_{k-1} pour évaluer s. Après l'optimisation, le joueur 1 envoie la meilleure valeur de X_k au joueur 2 qui va l'utiliser à la génération $k+1$ et le joueur 2 envoie la meilleure valeur de Y_k au joueur 1 qui va l'utiliser à la génération $k+1$. L'équilibre de Nash est atteint lorsque ni le joueur 1 ni le joueur 2 ne peut plus améliorer son critère.

En résumé, lorsque l'on minimise deux fonctionnelles $J_1(X,Y)$ et $J_2(X,Y)$, l'équilibre de Nash (X^*,Y^*) se traduit par :

$$X^* = \underset{X}{Argmin} \ (X,Y^*)$$

$$Y^* = \underset{Y}{Argmin} \ (X^*,Y)$$

On notera que dans le cas particulier où $J_1(X,Y) = J_2(X,Y) = J(X,Y)$ (un unique objectif), les conditions ci-dessus sont moins fortes que celle d'un optimum local car elles concernent les fonctions partielles $J(X,Y^*)$ et $J(X^*,Y)$ seulement.

3 Application à l'optimisation de profils grâce à l'AG2D

Nous avons travaillé avec l'algorithme génétique développé principalement par Nathalie Marco (l'AG2D).

3.1 Modélisation de l'écoulement et discrétisation

On utilise le modèle simplifié du fluide parfait compressible non visqueux. On travaille avec les équations d'Euler 2D en régime permanent :

- modélisation:

Les équations d'Euler 2D peuvent s'écrire sous la forme conservative:

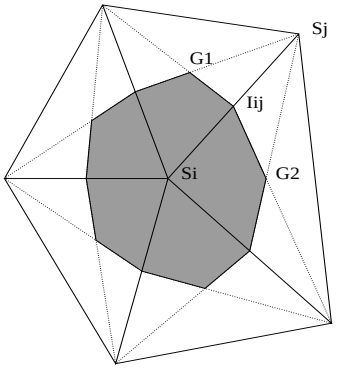
$$\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{F}(W) = 0 \quad ; \quad W = (\rho,\rho\vec{U},E)^T \quad ; \quad \vec{\nabla} = (\frac{\partial}{\partial x},\frac{\partial}{\partial y})^T \qquad (1)$$

où $\vec{F}(W) = (F_1(W),F_2(W))^T$ est le vecteur des flux convectifs, qui s'écrivent:

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 \\ \rho uv \\ u(E+p) \end{pmatrix} \quad ; \quad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

Avec ρ la masse volumique, $\vec{U} = (u,v)^T$ le vecteur vitesse, E l'énergie totale par unité de volume et p la pression obtenue par l'équation des gaz parfaits $p = (\gamma_p - 1)(E - \frac{1}{2}\rho\|\vec{U}\|^2)$ ($\gamma_p = 1.4$ le rapport des chaleurs spécifiques).

Soit Ω un domaine de calcul polygonal dans $\mathcal{R}^2$ et sa triangulation τ_h. Pour chaque sommet S_i, on définit un volume de contrôle C_i, construit en joignant le milieu du segment $[S_i,S_j]$ avec les centres de gravité des triangles entourant S_i, où $S_j \in N(S_i)$ et $N(S_i)$ les voisins de S_i (Figure ci-dessous). Le bord de C_i est notée ∂C_i, et la normale sortante unitaire de ∂C_i par $\vec{\nu}$.



La discrétisation spatiale utilise un schéma de volumes finis décentré. Pour chaque volume de controle C_i on résout n_i problèmes de Riemann 1D, n_i étant le nombre de voisins de S_i.

En intégrant les équations (**??**) sur C_i nous obtenons :

$$\begin{aligned} \iint_{C_i} \frac{\partial W}{\partial t}\vec{x} \;+\; \sum_{j\in N(S_i)} \int_{\partial C_{ij}} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma & \quad < 1 > \\ + \int_{\partial C_i \cap \Gamma_w} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma \;+\; \int_{\partial C_i \cap \Gamma_\infty} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma = 0 & \quad < 2 > \end{aligned} \tag{2}$$

où $\partial C_{ij} = \partial C_i \cap \partial C_j$, Γ_w et Γ_∞ les surfaces et la limite infinie. On discrétise au premier ordre par volume finis $< 1 >$:

$$< 1 > = \; W_i^{n+1} - W_i^n + \Delta t \sum_{j\in N(S_i)} \Phi_{\mathbf{F}}(W_i^n, W_j^n, \vec{\nu_{ij}}) \tag{3}$$

où $\Phi_{\mathbf{F}}$ représente une fonction de flux numérique tel que:

$$\Phi_{\mathbf{F}}(W_i, W_j, \vec{\nu_{ij}}) \approx \int_{\partial C_{ij}} \vec{\mathbf{F}}(W).\vec{\nu_i}d\sigma \quad , \quad \vec{\nu_{ij}} = \int_{\partial C_{ij}} \vec{\nu_i}d\sigma \tag{4}$$

Le décentrage est introduit dans l'équation **??** en utilisant un solveur de Riemann [9], produisant ainsi $\Phi_{\mathbf{F}}$ comme suit:

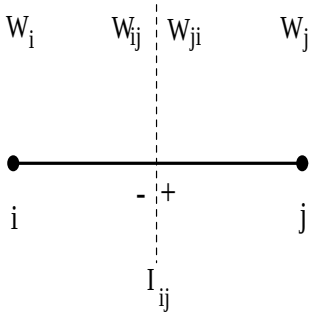
$$\Phi_{\mathbf{F}}(W_i, W_j, \vec{\nu_{ij}}) = \frac{\vec{\mathbf{F}}(W_i) + \vec{\mathbf{F}}(W_j)}{2}.\vec{\nu_{ij}} - \mid \mathbf{A}_R(W_i, W_j, \vec{\nu_{ij}}) \mid \frac{(W_j - W_i)}{2} \tag{5}$$

où $\mathbf{A}_R$ est la valeur principale de Roe de la matrice Jacobienne du flux $\dfrac{\partial \vec{\mathbf{F}}(W)}{\partial W}.\vec{\nu}$. On atteint le second ordre dans l'équation **??**, aprés une interpolation des états W_{ij} et W_{ji} à l'interface ∂C_{ij} (Figure ci-dessous) :

$$\tilde{W}_{ij} = \tilde{W}_i + \frac{1}{2}(\vec{\nabla}\tilde{W})_i.S_i\vec{S}_j \quad , \quad \tilde{W}_{ji} = \tilde{W}_j - \frac{1}{2}(\vec{\nabla}\tilde{W})_j.S_i\vec{S}_j \tag{6}$$

où $\tilde{W} = \left(\rho \ , \vec{U} \ , p\right)^T$. Un gradient moyen $(\vec{\nabla}\tilde{W})_i$ est obtenu aux nœuds en moyennant le gradient P_1-Galerkin sur chaque triangle de C_i. Finalement, la procédure de limitation de Van Albada est introduite dans l'interpolation (**??**) pour préserver la monotonie de l'approximation.

Ci-dessous : états physiques W_{ij} et W_{ji} interpolés sur une frontière S_i,S_j



- <u>intégration temporelle</u> :

Comme nous l'avons précisé, nous cherchons l'état stationaire de l'écoulement. Pour cela nous utilisons un schéma implicite permettant d'utiliser de grands pas temporels. En supposant que $W(\vec{x},t)$ est constant sur les C_i nous obtenons:

$$\mathrm{vol}(C_i)\frac{dW_i}{dt} + \Psi(W)_i = 0 \quad , \quad i = 1, \cdots, N_v \tag{7}$$

où $W_i = W(\vec{x_i},t)$, N_v le nombre de volumes de controle, et:

$$\Psi(W)_i = \sum_{j \in N(S_i)} \Phi_{\mathbf{F}}(W_{ij},W_{ji},\vec{\nu_{ij}}) + \int_{\partial C_i \cap \Gamma} \vec{\mathbf{F}}(W).\vec{\nu_i} d\sigma \tag{8}$$

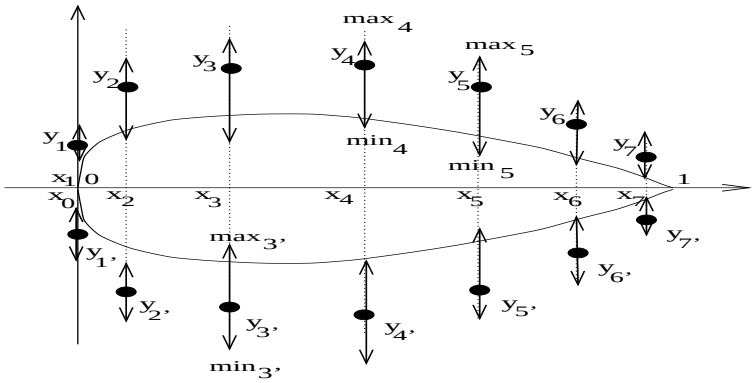
avec $\Gamma = \Gamma_w \cup \Gamma_\infty$. En appliquant une linéarisation au flux $\Psi(W^{n+1})$, nous obtenons la formulation quasi-Newton:

$$\left(\frac{\mathrm{vol}(C_i)}{\Delta t^n} + J(W^n)\right)(W^{n+1} - W^n) = -\Psi(W^n) \tag{9}$$

où $J(W^n)$ la matrice Jacobienne associée. A chaque pas d'intégration, le système linéaire est résolu en utilisant une méthode de relaxation de Jacobi.

3.2 Paramétrisation du profil

La paramétrisation du profil se fait par utilisation de 2 courbes de Bézier, pour nous d'ordre n=8, une pour représenter l'extrados, l'autre pour représenter l'intrados.



Ces courbes sont définie à l'aide de 9 points de contrôle de coordonnées (x_i,y_i) de la manière discrète suivante:

$$x(t) = \sum_{i=0}^{n} C_n^i t^i (1-t)^{n-i} x_i, \quad y(t) = \sum_{i=0}^{n} C_n^i t^i (1-t)^{n-i} y_i$$

où $C_n^i = \frac{n!}{i!(n-i)!}$ et t est le paramètre dont la valeur appartient à [0;1]. Ces formules sont évaluées en 80 points du maillage sur l'extrados et l'intrados. Afin de réduire l'espace de travail de l'algorithme génétique, nous ne permettrons aux points de contrôle de varier que dans la direction verticale, c'est-à-dire que les x_i sont fixés, aux valeurs suivantes:

$x_0 = x_1 = 0.00$, $x_2 = 0.15$, $x_3 = 0.30$, $x_4 = 0.45$, $x_5 = 0.60$, $x_6 = 0.75$, $x_7 = 0.90$, $x_8 = 1.00$.

Remarque : cet ensemble des x_i uniformément répartis dans l'intervalle [0;1] n'est pas le plus intéressant pour la paramétrisation des profils. On pourra

avantageusement utiliser une distribution en cosinus (distribution de Gauss-Lobato)

Les 2 premiers points de contrôle sont tous deux d'abscisse nulle ($x_0 = x_1 = 0$), ce qui contraint le profil à avoir une tangente verticale à l'avant. En effet les courbes de Bézier sont tangentes aux segments formés par les 2 premiers et les 2 derniers points de contrôle, le premier et le dernier point de contrôle appartiennent à la courbe. Ces 2 points (les extrémités) sont fixes, donc les ordonnées y_0 et y_8 sont fixes: $y_0 = y_8 = 0.00$. Ainsi nous allons jouer sur 14 paramètres pour déformer le profil: (y_1,y_2,y_3,y_4,y_5,y_6,y_7) pour l'extrados et ($y_{7'}$,$y_{6'}$,$y_{5'}$,$y_{4'}$,$y_{3'}$,$y_{2'}$,$y_{1'}$) pour l'intrados. De plus, on fixe les intervalles de variation suivants (pour une longueur de profil unité) :

- pour y_1 : [0.01; 0.03]
- pour y_2, y_3, y_4, y_5, y_6, y_7 : [0.01; 0.10]
- pour $y_{7'}$, $y_{6'}$: [0.000; 0.005]
- pour $y_{5'}$, $y_{4'}$, $y_{5'}$, $y_{2'}$, $y_{1'}$: [-0.10; -0.01]

3.3 Spécificités de l'AG2D

L'algorithme utilisé est un algorithme génétique mono-objectif de type minimisation (on cherche à minimiser la fonction objectif). Nous utilisons différentes fonctions objectifs selon le problème sur lequel on travaille. Par exemple on peut chercher à diminuer la traînée tout en conservant une valeur suffisante de la portance. Pour cela on prend comme profil modèle le RAE2822, on calcule ses valeurs de Cd et de Cl ; Cl est alors considéré comme la portance cible et on peut décrire la fonction d'adaptation comme suit :

$$f(profil) = Cd + 10(Cl - Cl^{cible})^2$$

Ici, on cherche à optimiser le profil, on a donc des populations de profils et chaque profil est décrit par 14 variables. De plus l'angle d'attaque est aussi considéré comme un paramètre à optimiser, on a donc 15 paramètres pour décrire chaque individu. Ces paramètres vont être codés en binaire, la longueur totale du chromosome (y_1,y_2,y_3,y_4,y_5,y_6,y_7,$y_{7'}$,$y_{6'}$,$y_{5'}$,$y_{4'}$,$y_{3'}$,$y_{2'}$,$y_{1'}$,α) est de 294 bits.

A chaque fois que nous utilisons l'algorithme génétique, nous gardons fixées un certain nombre de constantes et de procédures. En effet, des études ont été menées auparavent afin de déterminer certains paramètres comme les probabilités de croisement et de mutation afin que la convergence de l'algorithme génétique soit optimale. Nous avons utilisé les résultats obtenus, c'est-à-dire:

- type de sélection : roulette
- type de croisement : croisement simple
- probabilité de croisement : 0.9
- type de mutation : mutation binaire
- probabilité de mutation : 0.04
- procédure élitiste
- procédure «niche» avec $\sigma_{share} = 1.2$
- pénalisation des individus dont l'épaisseur maximale n'est pas comprise entre ± 10
- codage binaire des paramètres, précision: 10^{-6}
- cfl maxi: 1.10^6

On utilise un maillage à 14747 noeuds (160 sur le profil) et 29054 triangles. On travaille avec un Cluster de PC bipro Pentium III, on utilise 16 processeurs : parallélisation de l'AG (on traite 2 individus à la fois) et le domaine de maillage est divisé en 8 sous-domaines.

4 Résultats

4.1 Optimisation mono critère

4.1.1 Convergence de l'AG

Cette première étude avait pour but de déterminer de quelle façon l'AG2D converge en fonction du nombre d'individus et du nombre de générations, afin de déterminer par exemple si à nombre de calculs égaux, il était plus intéressant de travailler avec plus d'individus ou plus de générations. Pour cela il suffit de comparer les valeurs de la fonction objectif, par exemple lorsque l'on a travaillé avec 30 individus et 40 générations puis avec 40 individus et 30 générations.

La fonction objectif utilisée est celle pour laquelle l'AG2D a été initialement conçu :

$$f(profil) = Cd + 10(Cl - Cl^{cible})^2$$

Les valeurs de cette fonction, multipliées par un facteur 100 sont reportées dans le tableau ci-dessous; on a travaillé avec des nombres d'individus variant entre 15 et 50, des nombres de générations variant entre 5 et 50.

ind. / *gen.*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
5	4.1546	3.3637		0.9734	0.6932	2.0567	2.0013	1.3067
10	3.4261	1.2330		0.8794	0.5798	1.2002	1.9591	0.6806
15	3.4043	1.1309		0.7626	0.5761	0.9202	1.2295	0.6054
20	2.6962	1.1058		0.5304	0.5716	0.7893	1.0951	0.6046
25	2.5377	1.0537		0.5132	0.5653	0.7183	1.0413	0.5998
30	2.5100	0.9636		0.5065	0.5649	0.6385		0.5996
35	2.4668			0.5050	0.5636	0.6159		0.5978
40	2.4614			0.5039	0.5628	0.6091		0.5976
45	2.4454			0.5031	0.5581	0.6088		0.5975
50	2.3974			0.5025	0.5573	0.6084		0.5972

On constate tout d'abord que l'exécution du programme ne se réalise pas avec 25 individus et s'arrête au bout de 30 générations lorsque l'on a 20 individus, au bout de 25 lorsque l'on a 45 individus. Ceci est sans doute dû à des problèmes de robustesse, peut-être dûs à notre utilisation particulière. Par contre on constate que les individus sont effectivement de mieux en mieux adaptés au milieu puisque les valeurs des fonctions objectif décroissent avec les générations (procédure élitiste). Mais on ne peut pas expliquer pourquoi les résultats sont moins bons avec 40 individus qu'avec 30 puisque le nombre de calculs est plus élevé avec 40 individus...

Ainsi cette étude n'est pas concluante sur la convergence de l'AG2D, il semblerait néanmoins qu'il soit plus avantageux de mettre un grand nombre de générations au détriment du nombre d'individus.

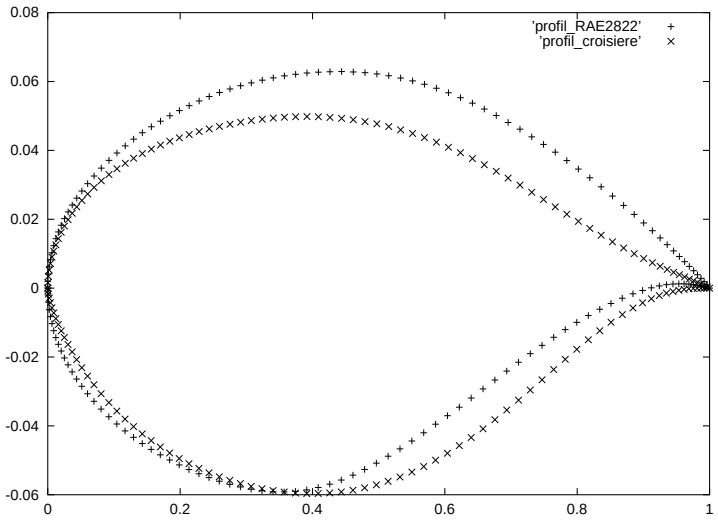
4.1.2 Optimisation de la traînée en régime de croisière

Nous considérons que le régime de croisière se situe à un nombre de Mach de 0.77 et à une incidence de 1 degré, ces conditions correspondant à des cas tests du groupe européen ECARP [10]. Dans ce cas précis la portance de l'aile n'est pas dimensionnante et le concepteur cherche à diminuer la consommation de carburant. Celle-ci étant inversement proportionnelle à la traînée, nous prendrons simplement comme fonction objectif (nous sommes dans le cas d'une minimisation) :

$$f(profil) = Cd$$

On a fixé l'angle d'attaque $\alpha = 1$ degré pour simplifier. Pour cela soit on passe à un algorithme génétique à 14 paramètres, soit on lui assigne un très faible intervalle de varation, par exemple [0.999, 1.001]. Nous avons choisi cette deuxième méthode qui était plus facile à mettre en place.

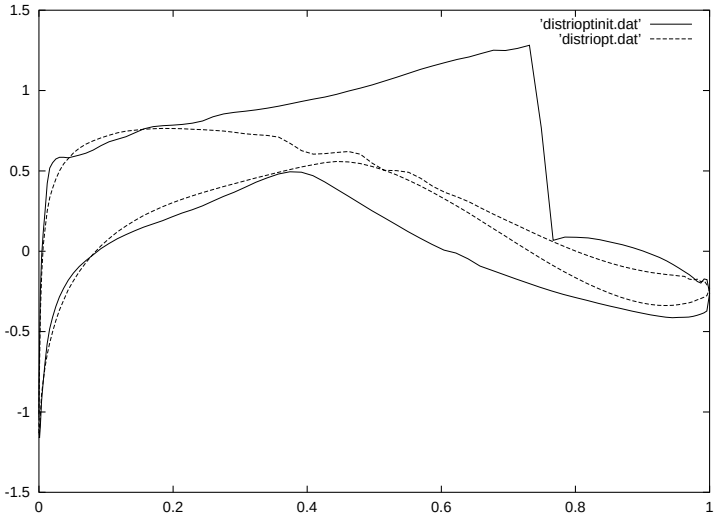
Nous avons ici travaillé avec 30 individus et 30 générations. L'épaisseur maximale ne devait pas être inférieure à 0.9*épaisseur du RAE2822 (sinon on aurait obtenu un profil très fin). Voici le profil obtenu, comparé avec celui du RAE2822 :



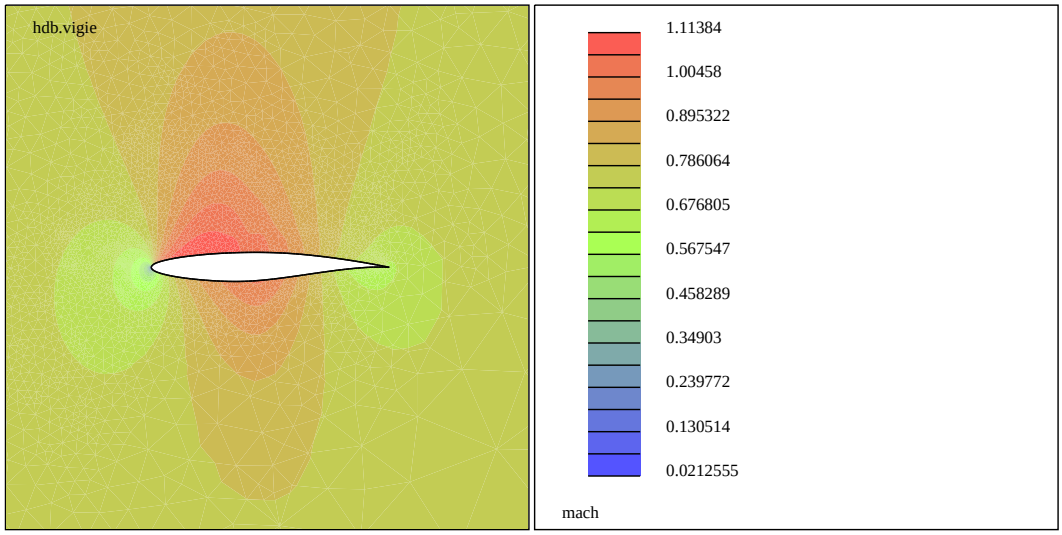
A ce régime, la portance et la traînée du RAE2822 valent respectivement $Cl = 0.682206$ et $Cd = 0.019135$

Le profil obtenu est de plus faible traînée (sa portance a elle-aussi diminué): $Cl = 0.257449$ et $Cd = 0.001354$.

Voici la courbe des pressions sur le profil; on remarque la chute brutale correspondant au choc sur la courbe du RAE2822. Le profil optimisé a amorti ce choc d'où la réduction de la traînée:



Ci-dessous, tracé des iso-Mach: les vitesses sur le dessus de l'aile sont progressivement décellérées.



Cette expérience ne constitue pas une optimisation très réaliste, car le problème est sous-contraint, mais permet de confirmer la bonne tendance de l'algorithme d'optimisation dans ce cas de calcul.

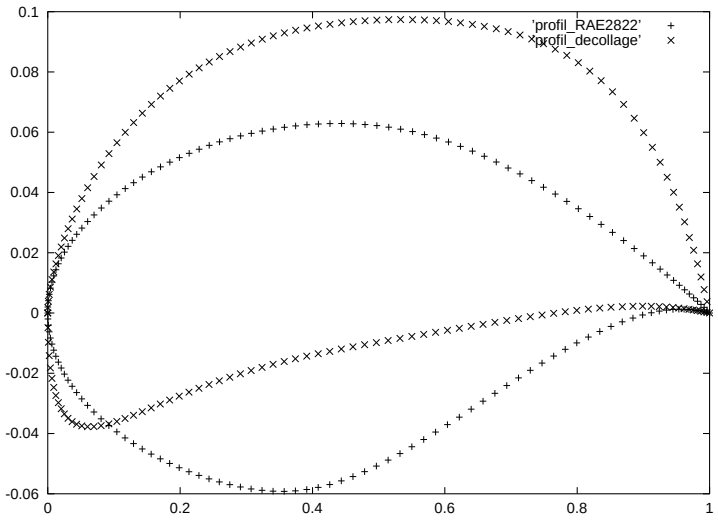
4.1.3 Optimisation de la portance en régime de décollage

Pour se conformer aux cas tests d'ECARP, nous considérons que le régime de décollage se situe à un nombre de Mach de 0.2 et à une incidence de 10.8 degré. Dans ce cas précis c'est la portance de l'aile qui requiert toute notre attention, nous pouvons nous désintéresser de la valeur de la traînée pendant la durée du décollage. Notre fonction d'adaptation devant être positive (et Cl ne dépassant jamais 2.8) et inversement proportionnelle à la portance, nous prendrons la fonction objectif suivante:

$$f(profil) = (Cl - 2.8)^2$$

On a fixé l'angle d'attaque $\alpha = 10.8$ degré. Pour cela nous lui avons assigné un très faible intervalle de variation: [10.799, 10.801].

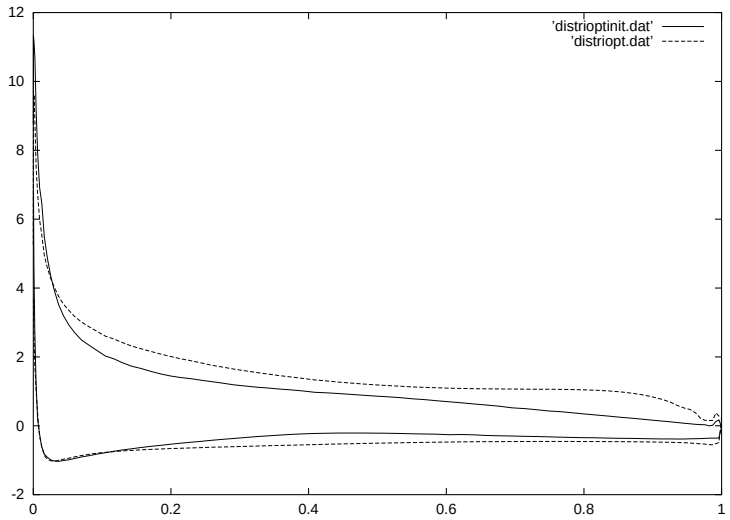
Nous avons ici travaillé avec 30 individus et 50 générations (conditions plus difficiles pour trouver les premiers profils). L'épaisseur maximale ne devait pas être supérieure à 0.9*épaisseur du RAE2822. Voici le profil obtenu, comparé avec celui du RAE2822:



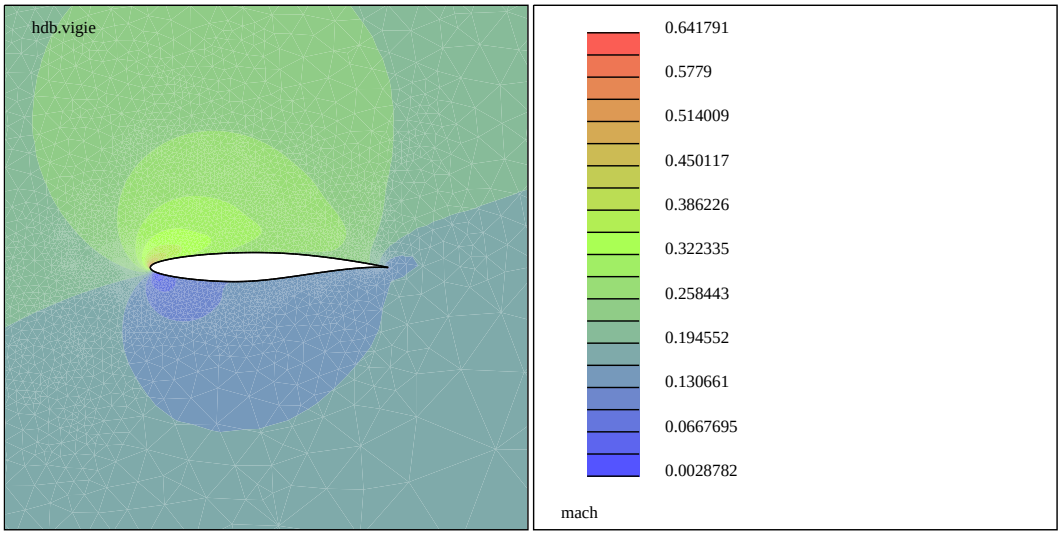
A ce régime, la portance et la traînée du RAE2822 valent respectivement $Cl = 1.505203$ et $Cd = 0.026608$.

Le profil obtenu est de plus forte portance (de plus forte traînée aussi): $Cl = 2.126513$ et $Cd = 0.045282$.

Voici la courbe des pressions sur le profil ; il n'y a plus de choc à cette faible vitesse.



Ci-dessous tracé des iso-Mach: les vitesses sont faibles à l'avant de l'intrados, on a donc de grandes pressions, donc une grande portance.



4.2 Optimisation de deux critères : équilibre entre le profil pour le régime de croisière et celui pour le décollage

4.2.1 Mise en place du jeu de Nash

Nous cherchons un profil réalisant un compromis entre les 2 profils solutions obtenus précédemment, c'est-à-dire un profil qui satisferait au mieux à la fois la condition faible traînée à M=0.77 et α=1 degré et la condition grande portance à M=0.2 et α=10.8 degré. Pour cela on va utiliser la stratégie de Nash avec 2 joueurs:

- Le premier joueur travaille sur une partie du profil, l'autre restant fixe, déterminée par le joueur 2 à la génération de Nash précédente. Il cherche un profil minimisant la traînée (toujours avec la contrainte sur l'épaisseur de $\pm$10%). Il travaille avec une population de 30 individus et on l'arrête au bout de 3 générations.

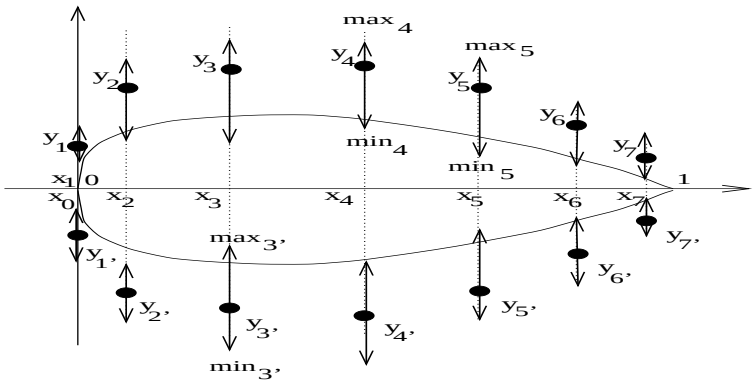
- Le deuxième joueur travaille sur l'autre partie du profil (la première restant fixe, déterminée par le joueur à la génération de Nash précédente). Il cherche un profil maximisant la portance (toujours avec la contrainte sur l'épaisseur de $\pm$10%). Il travaille sur une population de 50 individus et on l'arrête au bout de 3 générations.

4.2.2 Distribution des points de contrôle

Nous avons utilisé deux stratégies différentes pour la répartition des points de contrôle entre les 2 joueurs:

Une première stratégie consiste en la répartition alternée entre les 2 joueurs :

Le premier joueur travaille avec 8 variables: il travaille sur les ordonnées y_1, y_3, y_5, y_7, $y_{7'}$, $y_{5'}$, $y_{3'}$, $y_{1'}$ en gardant fixes les ordonnées y_2, y_4, y_6, $y_{6'}$, $y_{4'}$, $y_{2'}$. Inversement le deuxième joueur travaille avec 6 variables: il travaille sur les ordonnées y_2, y_4, y_6, $y_{6'}$, $y_{4'}$, $y_{2'}$ en gardant fixes les ordonnées y_1, y_3, y_5, y_7, $y_{7'}$, $y_{5'}$, $y_{3'}$, $y_{1'}$. Ces notations correspondent à la figure suivante:



En fait, cette stratégie s'est révélée infructueuse car elle ne nous permet pas de faire jouer le deuxième joueur. En effet une fois qu'il a remplacé ses y_1, y_3, y_5, y_7, $y_{7'}$, $y_{5'}$, $y_{3'}$, $y_{1'}$ par ceux trouvés par le joueur 1 à la première gńération, il n'arrive pas, même avec 50 individus, à trouver des profils cohérents pour le régime du décollage.

On a donc eu recours à une deuxième stratégie bien plus intéressante.

Cette deuxième stratégie a été envisagée après avoir remarqué que le travail effectué sur l'aile pour en augmenter la portance portait essentiellement sur l'avant de l'aile et que le travail de réduction de la traînée, lui, consiste en un raffinement de la partie arrière. Ainsi il est plus astucieux de faire jouer le joueur 1 (régime de croisière) sur les points de contrôles situés à l'arrière de l'aile, et le joueur 2 (décollage) sur les points situés sur l'avant de l'aile. Ainsi on donne au joueur 1 les ordonnées y_4, y_5, y_6, y_7, $y_{7'}$, $y_{6'}$, $y_{5'}$, $y_{4'}$ en gardant fixes les ordonnées y_1, y_2, y_3, $y_{3'}$, $y_{2'}$,$y_{1'}$. Et inversement, le joueur 2 travaille sur les ordonnées y_1, y_2, y_3, $y_{3'}$, $y_{2'}$,$y_{1'}$ en gardant fixes les ordonnées y_4, y_5, y_6, y_7, $y_{7'}$, $y_{6'}$, $y_{5'}$, $y_{4'}$.

Avec cette deuxième stratégie, on aboutit à un équilibre (les joueurs ne peuvent plus améliorer le profil) en 5 générations de Nash.

4.2.3 Solution du jeu de Nash

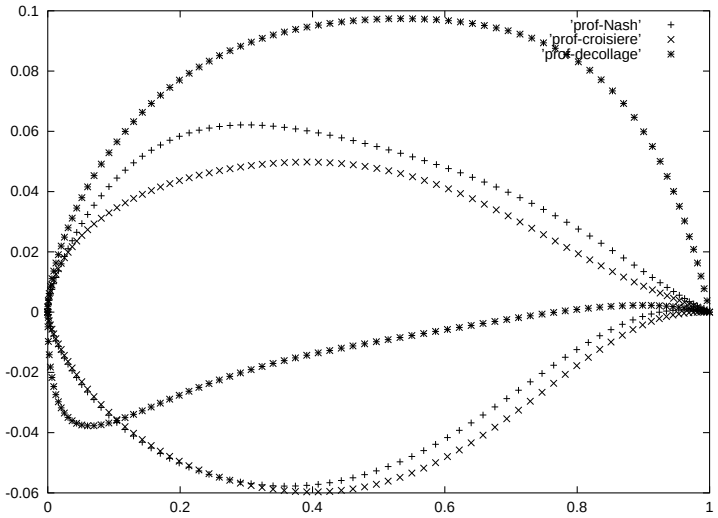
Voici la description du profil obtenu :

Tout d'abord, pour information, voici les ordonnées des points de contrôle de la solution:

y_1 =0.01663798, y_2 =0.089088935, y_3 =0.0888282669, y_4 =0.0275687862, y_5 =0.0718557471,

y_6 =0.0389276304, y_7 =0.0102147486, $y_{7'}$ =0.00329785611, $y_{6'}$ =0.00447493706, $y_{5'}$ =-0.0758793219, $y_{4'}$ =-0.0686219012, $y_{3'}$ =-0.0559501895, $y_{2'}$ =-0.0743608898, $y_{1'}$ =-0.0141032258

Ci-dessous, le tracé du profil comparé avec ceux obtenus au 3.2 et au 3.3:



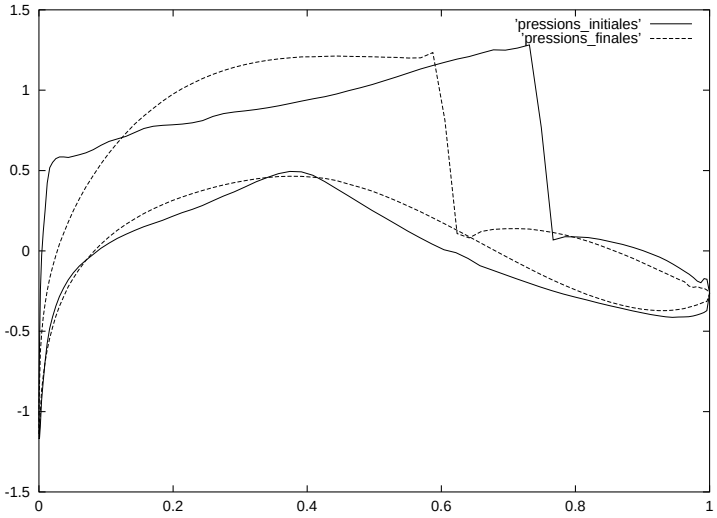
On peut remarquer que le profil solution essaie de coller à l'avant avec le profil pour le décollage, à l'arrière avec le profil solution pour le régime de croisière.

Le profil obtenu a pour portance et pour traînée $Cl = 0.506858$ et $Cd = 0.020330$ en régime de croisière et $Cl = 1.429557$ et $Cd = 0.062452$ dans le cas du décollage. On récapitule ci-dessous les valeurs de portance et de traînée :

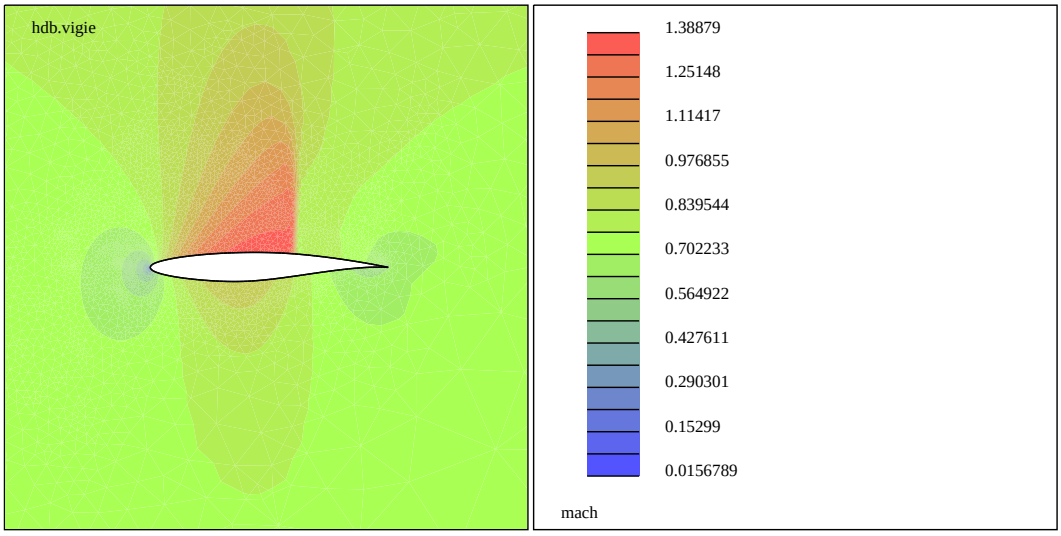
Croisière	Cl	Cd
mono critère 1	0.257449	0.001354
mono critère 2	1.319319	0.052277
multi critère	0.506858	0.020330
Décollage	Cl	Cd
mono critère 1		
mono critère 2	2.126513	0.045282
multi critère	1.429557	0.062452

Lorsque l'on se place dans la situation de croisière, on constate que le profil multi critère ne donne pas une traînée aussi faible que le mono critère 1 (solution obtenue au 4.1.2), mais il est quand même bien meilleur que le mono critère 2 (solution obtenue au 4.1.3). De même, dans le cas du décollage, le profil multi critère n'a pas une portance aussi élevée que le profil mono critère 2. Nous n'avons pas pu évaluer la portance et la traînée du mono critère 1 dans les conditions de décollage car il est trop inadapté donc toujours rejeté.

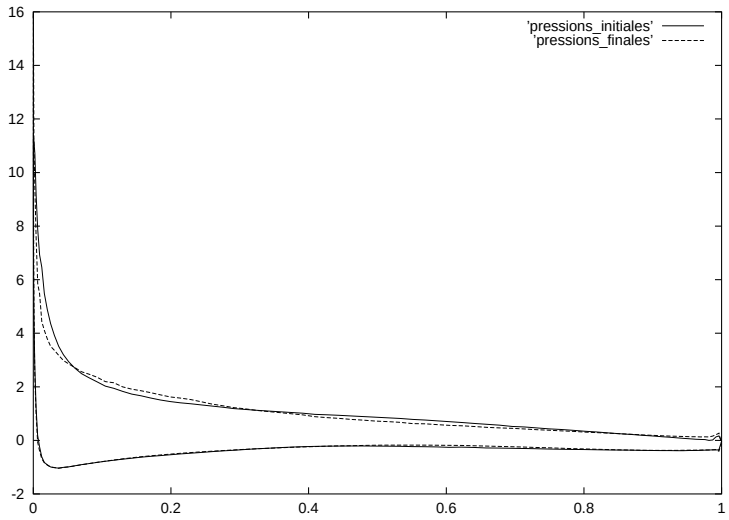
Profil des pressions en régime de croisière :



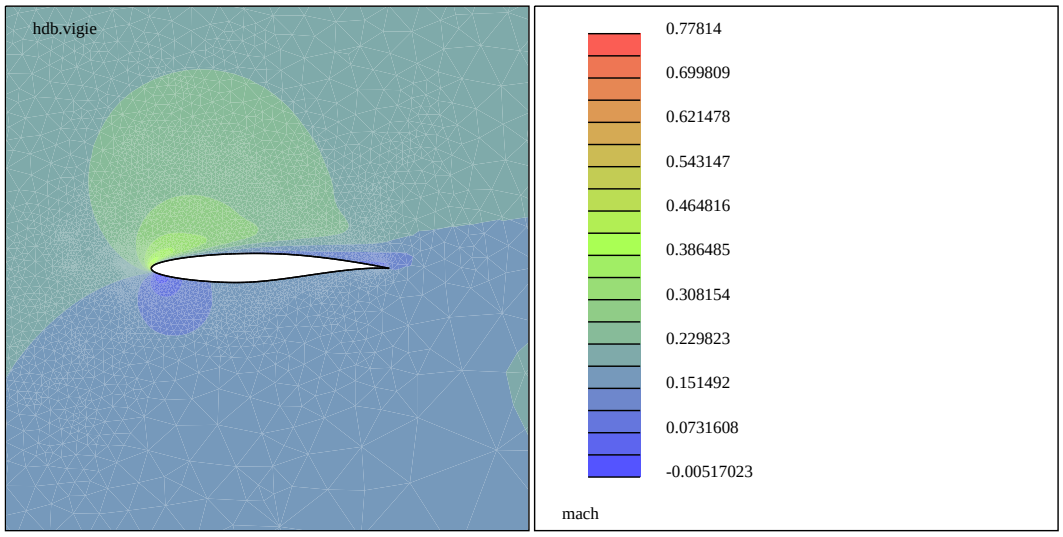
Iso-Mach du profil solution en régime croisière :



Profil des pressions en régime décollage :



Iso-Mach du profil solution en régime décollage :



Ces graphes nous permettent de constater que le profil solution n'est pas aussi satisfaisant pour chaque critère que les profils trouvés au 4.1.2 et au 4.1.3.

En effet le choc est beaucoup moins bien amorti par exemple donc la traînée beaucoup plus élevée en régime de croisière, et les pressions sont beaucoup plus basses en régime de décollage. Néanmoins ses caractéristiques, lorsque l'on doit prendre les 2 critères en considération, sont bien meilleures que celles des profils mono critères. On a donc bien réalisé un compromis.

Conclusions

Dans cette étude, on a réalisé des expériences d'optimisation de forme de profil d'aile d'abord selon un seul critère (minimisation de la traînée dans les conditions de la croisière, ou maximisation de la portance dans les conditions du décollage). Ces expériences ont été réalisées au moyen du code AG2D mise au point antérieurement par N. Marco [3].

Puis on a réalisé une optimisation de forme en mettant en concurrence deux objectifs simultanément. Pour cela, on a utilisé une stratégie de Nash. Un équilibre a été atteint dans le cas d'une répartition avant-arrière des territoires entre les 2 joueurs. Par contre, une expérience préalable avec une répartition alternée des points de contrôle, a échoué.

On conclut donc que si la théorie des jeux (jeux coopératifs : équilibre de Pareto ; jeux compétitifs : équilibre de Nash ; stratégie maître-esclave : théorie de Stackelberg, etc) fournit un cadre particulièrement souple pour traiter des problèmes d'optimisation multi objectifs (ou multi disciplinaires), l'identification intelligente des territoires reste un problème ouvert pour la recherche.

Annexe : Quelques éléments pour utiliser l'AG2D

Cette annexe vise simplement à donner quelques éléments au futur utilisateur de l'AG2D. On ne prétend pas ici donner un shéma détaillé du code, mais simplement le minimum nécessaire. Dans un premier temps, nous allons expliquer comment est structuré le programme en reprenant le nom des principales routines. Nous nous attacherons ensuite à l'aspect purement pratique de son utilisation, c'est-à-dire que nous détaillerons les entrées, les sorties et les commandes.

Squelette du code

L'AG2D est constitué de 5 sous-répertoires : EXE, SAUV, SRC-AG, SRC-FLOW, SRC-PARAL. Les 3 derniers ont pour rôle de séparer les routines sources liées au fonctionnement de l'algorithme génétique, celles liées aux calculs d'écoulements et celles utiles pour la parallélisation de l'AG. Ces routines sont codées en Fortran 77. Dans SAUV se trouve l'ensemble des fichiers, dans EXE les fichiers entrées et sorties.

Le squelette de l'AG se trouve dans SRC-AG/Algogen.f. On commence par calculer l'écoulement autour du profil modèle qui est celui du RAE2822. Pour cela on fait appelle à FirstFluid.f

- FirstFluid.f va calculer pour la première fois un écoulement. Pour celà, il commence par lire les données relatives au maillage et à l'écoulement dans test.data par Donnees.f . Il effectue ensuite son maillage grâce aux subroutines SubMsh.f, MaillaNew.f, Aires.f, ClHaut.f, Seg2D.f, Init.f, Claprc.f, Etat.f (solveur des équations d'Euler ou de Navier-Stockes à 2 dimensions). On en tire alors la distribution de pression, ce qui nous permet de calculer les valeurs de la portance et de la traînée du RAE dans cet écoulement (PrCibl.f, DistriPres.f, Portance.f). Ces valeurs sont appelées portance cible et traînée initiale, elles sont stockées dans test.output.

Maintenant on se lance dans l'AG proprement dit en appelant :

- MainLoop.f : celui-ci commence par lire les données concernant les paramètres de l'algorithmes, données qui se trouvent dans entree.dat, par Idata.f. Ensuite il crée une première population en générant tous les individus au hasard, grâce à InitPop.f :

InitPop.f génère le chromosome de chacun des individus au hasard. Pour cela il utilise Agran.f (génère un chiffre au hasard dans [0;1]) puis Flip.f donne un 0 ou un 1 selon si ce chiffre est supérieur ou inférieur à 0.5. On fait cela pour toute la longueur de la chaîne représentant le chromosome de chaque individu. On traduit ensuite en réel la valeur du chromosome par Decode.f. Une fois que l'on a ainsi nos individus, on va calculer l'épaisseur du profil correspondant par CalcEp.f: l'épaisseur est définie comme la plus grande distance entre 2 points de même abscisse. Pour cela on fait appelle à ContrProf.f qui calcule à partir des 16 points de contrôle, 160 points du profil par la formule de Bézier. Si l'épaisseur d'un profil dépasse de $\pm 5\%$ de celle du RAE2822, on attribue à ce profil une valeur de fonction d'adaptation de 5 (c'est très grand donc l'individu sera pénalisé). Il s'agit maintenant d'évaluer ces individus: on les évalue 2 par 2 puis le dernier si on en a un nombre impair. On remaille (Movmeshp.f) puis on calcule la valeur de la fonction d'adaptation par Objfun.f. Cette routine calcule l'écoulement de la même manière dont cela a été fait dans FirstFluid.f puis il fait appelle à FnelleCout.f où est définie la fonction d'adaptation en fonction de la portance et de la traînée.

Statit.f: fait appelle à Statis.f qui calcule quelques statistiques comme le min, le max et la moyenne des valeurs des fonctions d'adaptation des individus de la population. On stocke la valeur la plus basse.

On ouvre ici une boucle sur les générations puis on lance

Generate.f: à chaque génération, on va faire les 3 opérations de base: la sélection, le croisement et la mutation, puis on effectue l'évaluation. Tout cela est fait en travaillant par paire d'individus. On commence par mélanger les chromosomes, c'est-à-dire qu'on redistribue au hasard les numéros des chromosomes grâce à la routine Mélange.f. Ensuite on ouvre une boucle sur la demi-population et selon le choix effectué dans entree.dat on lance la sélection par tournoi (Select.f) ou par roulette (Selectroul.f). Ensuite, toujours selon le choix lu dans entree.dat, on lance un croisement simple (Crossover.f) ou un croisement uniforme (CrossUnif.f). On décode ensuite les valeurs binaires des chromosomes. On effectue ensuite un test sur l'épaisseur avec ces nouvelles valeurs de chromosomes et on évalue les individus 2 par 2 exactement comme dans InitPop.f.

Statit.f

Elitism.f : on garde le meilleur individu trouvé ou on recopie celui de la génération précédente s'il est meilleur, ceci en binaire et en réel.

Si on a trouvé un meilleur individu, on le dessine avec ContrProf.f et on stocke ses coordonnées dans prof. Puis on écrit dans maxavgmin.dat et dans test.output les valeurs de la génération correspondante, du min, du max et de la moyenne des fonctions d'adaptations.

Lorsque l'on est arrivé à la fin des générations ou lorsque l'algorithme a convergé (cas ou le min de la fonction d'adaptation est devenu inférieure à 10^{-6}), on appelle

FinalFlow.f : cette routine calcule pour le meilleur individu de la dernière génération, le profil avec ContrProf.f, l'écoulement de la manière vue plus haut, la distribution de pression (DistriPress.f) et sa portance et sa traînée (Portance.f), qu'on affiche dans test.output.

Utilisation du code

Dans un premier temps nous allons voir comment lancer les calculs puis comment modifier les entrées et enfin comment exploiter les sorties.

Avant de lancer les calculs, il faut effacer tous les fichiers de sortie. Pour cela une commande clean se trouve sous EXE. Ensuite, pour lancer les calculs parallèle, il faut tout d'abord avoir l'environnement du Cluster puis se placer sur une des machines du Cluster ; il existe 2 ensembles de machines, les <<pf>> (les plus puissantes) et les <<galere>>. On peut décider de travailler sur un des 2 ensembles ci-dessus. Pour passer de l'un à l'autre il suffit de changer dans AG2D/EXE/job2D la ligne <<bsub -o ag-out -e ag-err -m linux-pf -n $1 mpijob Ag2D.out>> en <<bsub -o ag-out -e ag-err -m linux-galere -n $1 mpijob Ag2D.out>>. On peut se placer par exemple sur pf1 pour lancer l'AG2D. Pour cela on utilise la commande <<rsh pf1>>. Pour compiler le programme, se placer au niveau AG2D/ et lancer la commande <<gmake Ag2D.out>>. Une fois compilé, on peut lancer le code en se plaçant au niveau AG2D/EXE et en tapant <<job2D 16>>. On utilise alors 16 processeurs. Pour n'en utiliser que 8, il suffit de taper <<job2D 8>> en prenant soin de changer le 2 en 1 dans AG2D/EXE/config.data.

Si l'on souhaite vérifier que le programme tourne et visualiser sur quelles machines il tourne, il suffit de taper la commande <<bjobs>>. Si l'on souhaite stopper le programme on utilisera <<bkill>> suivi du numéro du job.

Maintenant si l'on veut changer les paramètres de l'AG, il faut modifier le fichier entree.dat ; le coef1 est le nombre de paramètres (14 y_i et l'angle d'attaque), les coeff2 sont les bornes inf des paramètres, les coef3 en sont les bornes sup (ces coefficients ne sont pas utilisés, en réalité on utilise les coef4), les coef4 sont définis avec les autres coefficients dont les coef5 qui sont les nombres de bits codant les paramètres: $coef4 = \frac{2^{coef5}-1}{coef3-coef2}$. Le coef6 est la somme des coef5 donc la longueur du chromosome. Puis on a le nombre d'individus, le nombre de générations, la probabilité de croisement, de mutation, la graîne pour la génération de nombres aléatoires, les 3 coefficients propres à la procédure sharing, le choix de la méthode de sélection, le choix de la méthode de croisement, 3 coefficients relatifs à l'hybridation.

Pour changer les paramètres de l'écoulement, on doit modifier le fichier AG2D/EXE/test.data ; par exemple on peut modifier le nombre de Mach ou l'angle de d'attaque mais il faut alors prendre soin de modifier aussi le fichier sol-init.data.

En ce qui concerne les fichiers de sortie et leur exploitation : on trouve dans test.output, les valeurs de la portance et de la traînée du RAE et celles du meilleur profil. D'autre part, maxavgmin.dat nous donne une idée sur la convergence de l'AG : on a pour chaque génération le minimum des valeurs prises par la fonction objectif. Dans liftdrag.dat on peut visualiser le nombre d'individus qui conviennent à l'écoulement à chaque génération. Les fichiers ag-out et ag-err permettent d'accéder aux messages d'erreur. Pour visualiser un profil, par exemple le trentième, et le comparer avec celui du RAE, on peut utiliser gnuplot et la commande <<plot 'peauRAE.data', 'prof-00030'>>. Il est interessant aussi de visualiser les iso-Mach ou les iso-P. Pour cela si l'on dispose de Vigie, il suffit de remplacer test.sol par AG2D/EXE/solution.data puis de faire <<MkVigie>> et <<vigie hdb.vigie>> et de se placer en <<multi2Dplot>>.

Références

[1] Stadler W., *Multi criteria Optimization in Engineering and in the Sciences*, Plenum Press (1998)

[2] Goldberg D.G., *Algorithmes génétiques, exploration, optimisation et apprentissage automatique.* Addison-Wesley (1989)

[3] Marco N., Désidéri J.-A., Lantéri S., *Multi-objective Optimization in CFD by Genetic Algorithms.* INRIA Sophia, RR-3686 (04/1999)

[4] Poloni C.. Hybrid QA for multi objective aerodynamic shape optimization. In G.Winter, J.Periau M.G. and Cuesta P. (eds) *Genetic Algorithms in Engineering and Computer Science*, John Wiley, (1995) p.397-415.

[5] Tang Z.L., Désidéri J.A., Périaux J., *Multi-objective Shape-Design Optimization and Inverse Problems using Control Theory and Nash Games.* EC-COMAS CFD 2001 Conference, Swansea, 4-7 sept. 2001.

[6] Nash J.F., Noncooperative games, *Annals of Mathematics*, p. 54-289

[7] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, ISBN: 3-540-55387-8 (1992)

[8] www.chez.com/produ/abdou

[9] Roe P.L., *Approximate Riemann solvers, parameters vectors and difference schemes.* In Journal of Computation Physics, 43 :357-371 (1981)

[10] Periaux J., Bugeda G., Chaviaropoulos P.K., Giannakoglou K., Lanteri S., Mantel B., *Optimum Aerodynamic Design & Parallel Navier-Stockes Computations.* ECARP-European Computational Aerodynamics Research Project, Notes on Numerical Fluid Mechanics, Volume 61, vieweg.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399