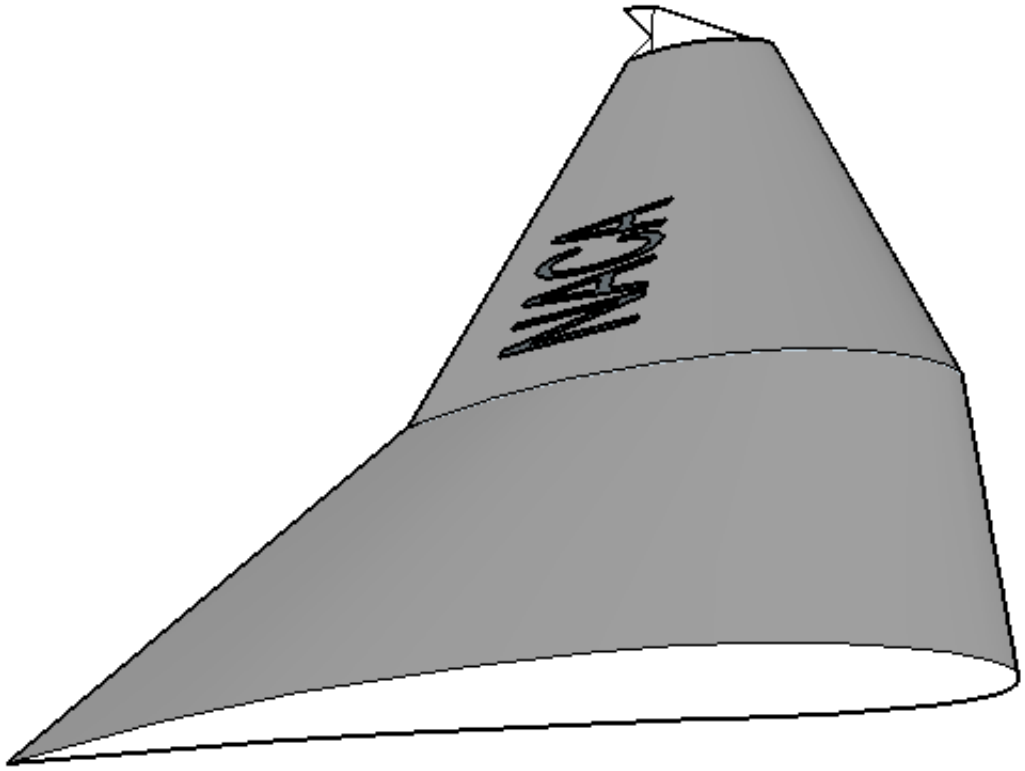
Projet Tuteuré
Minimisation de la trainée

May 28, 2015



Degoul Romain
Giammetta Nicolas

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Table des matières

1 Introduction **3**

2 Fonction DataTreatment **3**
2.1 Les profils NACA 3
2.2 Le logiciel Xfoil 5
2.3 La chaîne d'optimisation 6

3 Tests du logiciel Xfoil **7**

4 Optimisation du profil **11**
4.1 Maximisation de la portance 12
4.2 Présentation des profils de départ 13
4.3 Minimisation de la traînée (Sous contrainte CL=0.5) 15
4.4 Etude de la fonction DataTreatment 17
4.5 Minimisation de la traînée (Sous contrainte CL=0.5 et m=0.03) 20
4.6 Optimisation multi-objectifs 22

5 Conclusion **25**

6 Annexes **26**
6.1 Implémentation des algorithmes d'optimisation 26
6.1.1 Algorithme BFGS 26
6.1.2 Algorithme de Nelder-Mead 26
6.2 Données expérimentales de soufflerie 27

7 Références bibliographiques **31**

1 Introduction

Le but de notre bureau d'études est d'optimiser la forme d'une aile d'avion afin d'avoir une consommation en carburant la plus faible possible. On va utiliser une modélisation de cette aile sur laquelle on pourra calculer plusieurs fonctions objectifs comme la portance ou la trainée ce qui nous permettra de trouver une forme d'aile en fonction des critères que l'on veut remplir. Comme modélisation de l'aile d'avion, nous allons utiliser les profils NACA qui nous permettront de représenter des profils d'ailes en fonction de trois paramètres. Nous allons ensuite utiliser le logiciel Xfoil pour calculer les fonctions objectifs sur ces profils NACA. On s'intéressera notamment à la portance, la traînée et le moment de la résultante aérodynamique. Nous allons intégrer le calcul des fonctions objectifs dans un processus d'optimisation afin de trouver le profil NACA qui favorise le mieux un résultat des fonctions objectifs. On cherchera notamment les profils pour lesquels on obtient une traînée minimale pour une portance fixée. On commencera cet exposé par la création de la fonction Data-treatment qui associe à un profil NACA les résultats de Xfoil. On comparera ensuite les résultats de Xfoil avec des données réelles de soufflerie afin de s'assurer que la fonction Data-Treatment renvoie des données pertinentes. Pour terminer, on présentera les algorithmes d'optimisation que l'on a utilisés et on donnera les profils validant le mieux les résultats que l'on veux pour nos fonctions objectifs.

2 Fonction DataTreatment

Dans cette partie, nous allons introduire les différents outils que nous allons utiliser tout au long de ce projet.

2.1 Les profils NACA

Un profil NACA sert à décrire la forme d'une aile d'avion à l'aide de quatre chiffres MPXX.

- M est la cambrure maximale en pourcentage par rapport à la longueur de la corde
- P détermine la position de ce maximum de cambrure sur la corde. Il s'agit de la valeur de la position en dizaine de pourcents de la longueur de la corde à rapport au bord d'attaque.
- XX définissent l'épaisseur maximale du profil en pourcentage par rapport à la longueur de la corde.

Regardons l'influence de ces paramètres sur des exemples. Commençons par le profil NACA0015, on choisit donc une cambrure nulle et une épaisseur de 15% par rapport à la longueur. La cambrure étant nulle, le profil est symétrique selon un axe horizontal.

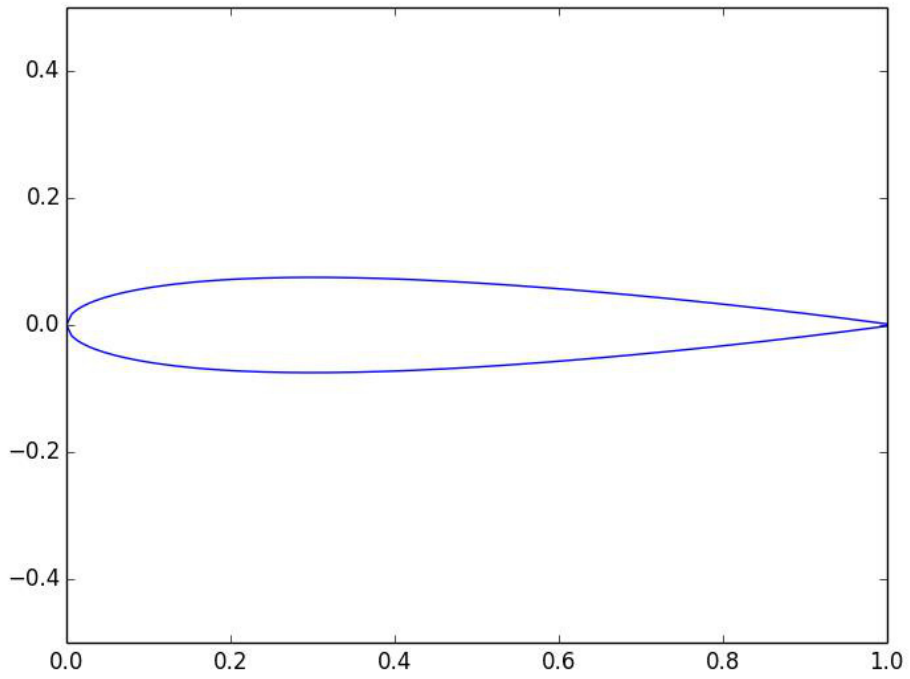


Figure 1: Profil NACA 0015

Prenons maintenant le profil NACA8415, on choisit cette fois-ci une cambrure égale à 8% de la corde et on la position de ce maximum sur l'axe horizontal à 40% de la longueur de la corde.

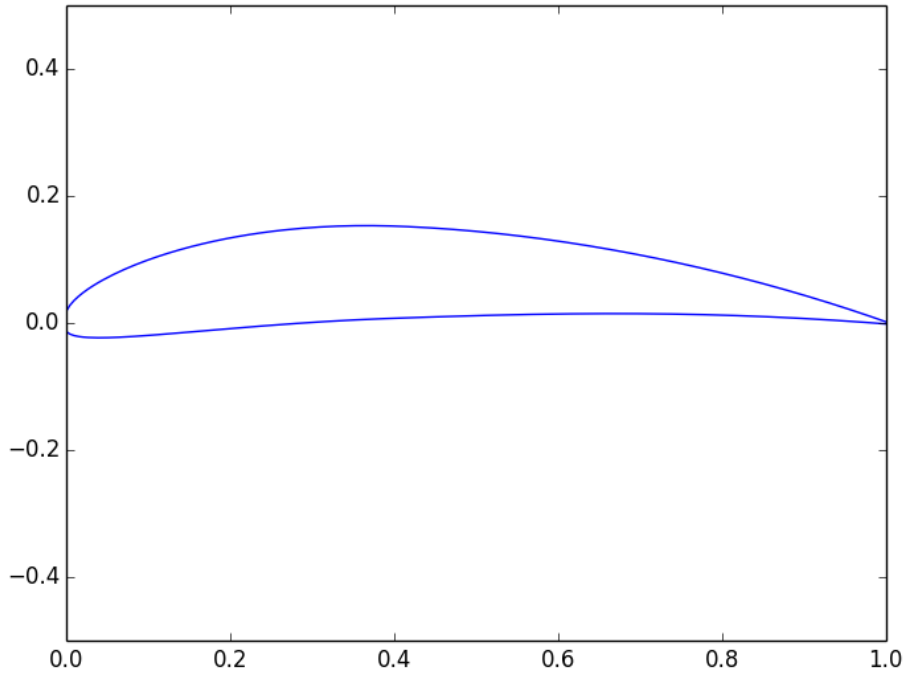


Figure 2: Profil NACA 8415

L'avantage de la modélisation de profil par le système NACA est que l'on peut directement choisir des paramètres très pertinents dans la conception d'une aile d'avion (cambrure, épaisseur).

Au niveau des équations mathématiques, un profil NACA se forme en deux temps. On définit d'abord l'expression de la demi-épaisseur d'une aile et ensuite on va donner l'expression mathématique de la ligne autour de laquelle on va reporter la demi-épaisseur. L'équation de la demi-épaisseur de l'aile est la suivante :

$$y_t = \frac{t}{0.2}c\left(0.2969\sqrt{\frac{x}{c}} - 0.1260\frac{x}{c} - 0.3516(\frac{x}{c})^2 + 0.2843(\frac{x}{c})^3 - 0.1015(\frac{x}{c})^4\right)$$

avec c la longueur de la corde du profil (on peut la fixer à 1) et t correspond à la valeur de XX en valeur réelle et non en pourcentage. On va remporter cette épaisseur autour de la courbe définie par l'équation :

$$y_c = \begin{cases} m\frac{x}{p^2}(2p - \frac{x}{c}) & \text{si } 0 \leq x < pc \\ m\frac{c-x}{(1-p)^2}(1 + \frac{x}{c} - 2p) & \text{si } pc \leq x \leq c \end{cases}$$

Au final, la paramétrisation d'un profil NACA est la suivante :

$$\begin{cases} x_U = x - y_t sin(\theta) & y_U = y_c + y_t cos(\theta) \\ x_L = x + y_t sin(\theta) & y_L = y_c - y_t cos(\theta) \end{cases}$$

où

$$\theta = arctan(\frac{dy_c}{dx})$$

$$\frac{dy_c}{dx} = \begin{cases} \frac{2m}{p^2}(p - \frac{x}{c}) \\ \frac{2m}{(1-p)^2}(p - \frac{x}{c}) \end{cases}$$

Comme notre but est de trouver une forme d'aile optimale selon certains critères, nous allons créer une chaîne de fonctions qui prendra en entrée les paramètres pour dessiner un profil NACA et donnera en sortie des informations sur la portance et la traînée. Dans la pratique, nous allons commencer par créer une fonction create qui prendra en arguments des coordonnées NACA et renverra deux vecteurs X et Y qui sont les coordonnées des points constituant le profil. Nous allons ensuite faire traiter le profil que l'on a choisi par le logiciel Xfoil. Xfoil est un programme qui permet de simuler des essais en soufflerie sur un profil NACA. Xfoil trace les polaires d'une aile choisie, c'est-à-dire l'expression des coefficients de portance et de traînée, CL et CD en fonction de l'angle d'incidence alpha.

2.2 Le logiciel Xfoil

Le logiciel Xfoil permet d'étudier des profils d'ailes d'avion en domaine subsonique, c'est-à-dire que les vitesses mises en jeu sont très inférieures à la vitesse du son (< 0.2 mac) ce qui permet d'éviter l'apparition d'onde de choc. Xfoil utilise l'hypothèse d'incompressibilité, c'est pourquoi on doit respecter le fait d'avoir de faibles vitesses.
Voici le script que nous utilisons avec Xfoil :
load NACA_Airfoil.txt % Airfoil charge le fichier des coordonnées du profil que l'on a créé avec la fonction create dans DataTreatment.py
pane % permet de se donner assez de points pour dessiner le profil. Ce nombre est habituellement de 140 mais il peut être modifié en fonction de la courbure du profil. (maximum 350 points)
OPER % permet d'entrer en mode opérateur
visc Re % permet d'entrer en mode visqueux et on donne un nombre de Reynolds
iter nit % permet de fixer le nombre d'itération que l'on veut pour les algorithmes de calculs
pacc % demande a Xfoil de sauvegarder les résultats obtenus dans le fichier result.txt
result.txt
aseq alpha0 alpha1 h % fixe l'échantillon de valeur d'angle d'incidence alpha que l'on veut pour calculer la polaire (de alpha0 à alpha1 par pas de h)
pacc
quit

À la place de **aseq**, on peut choisir les commandes **alfa** pour une unique valeur de alpha ou bien **CL** ou **Cseq** pour se fixer des valeurs de C_L.

Voici les résultats que l'on obtient lorsque l'on lance Xfoil à part avec ces commandes. Un tableau affiche pour chaque alpha les valeurs C_L, C_D, C_M.

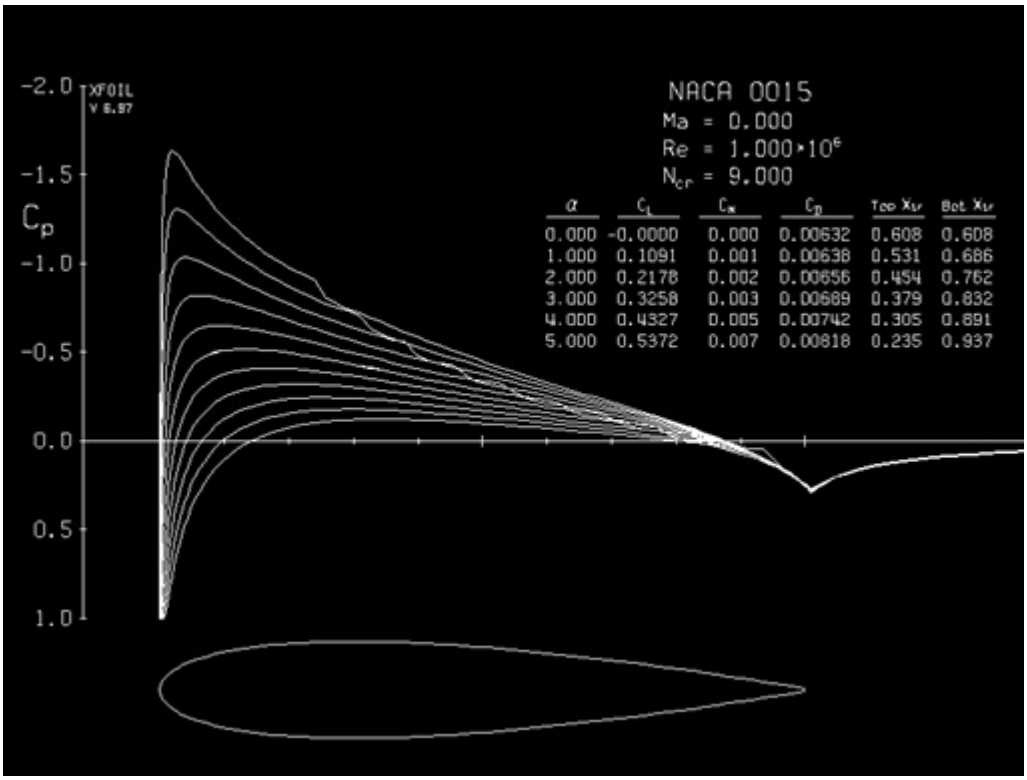


Figure 3: Résultats de Xfoil

C_L est un coefficient sans dimension lié à la portance (lift en anglais). De même, C_D est un coefficient sans dimension lié à la traînée (drag en anglais). La portance et la traînée sont les projections de la force totale s'exerçant sur l'aile. La portance est la force servant à faire voler l'avion et la traînée est la force s'opposant au mouvement de l'avion. C_M est le moment de la résultante aérodynamique. Top x_{tr} et bot x_{tr} sont les positions où se fait la transition régime laminaire/régime turbulent respectivement au niveau de l'extrados et au niveau de l'intrados.

Tous ces résulats sont issus de la modélisation interne de Xfoil et doivent donc être vérifiées avec des données réelles. Le logiciel se base sur une méthode de type potentiel. On se place dans la cadre de l'hypothèse d'incompressibilité et on néglige les effets visqueux. Dans ce cas, l'équation de conservation de la masse du système de Navier-Stokes devient :

$$div(\overrightarrow{v}) = 0$$

On considère également que le mouvement est irrotationnel donc

$$\overrightarrow{rot}(\overrightarrow{v}) = 0$$

Ce qui implique que la vitesse dérive d'un potentiel :

$$\overrightarrow{v} = \overrightarrow{grad}(\phi)$$

On obtient l'équation $\Delta\phi = 0$ que va résoudre Xfoil.

2.3 La chaîne d'optimisation

Une fois que nous sommes capables de calculer les coefficients qui nous intéressent à partir d'un profil NACA, on cherche à intégrer ces fonctions dans un processus d'optimisation pour que l'on puisse déterminer le meilleur profil possible pour des critères donnés sur la portance, la traînée et l'angle d'incidence.

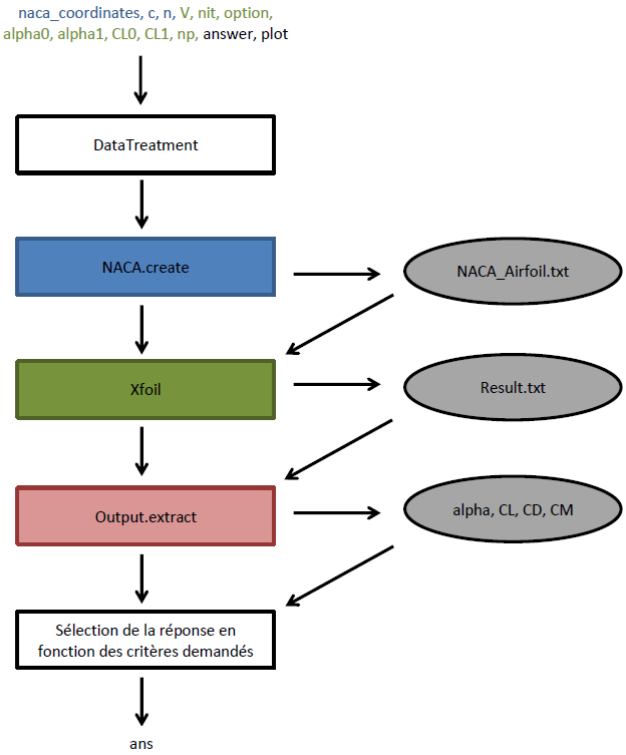


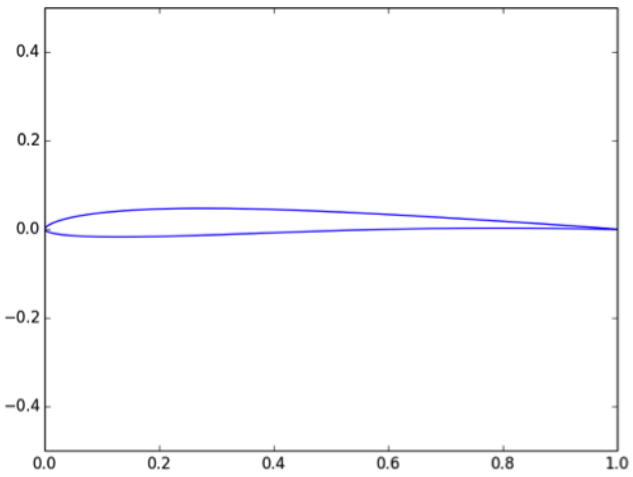
Figure 4: Processus de calcul des fonctions objectifs avec DataTreatment

Voici la fonction DataTreatment à intégrer dans un processus d'optimisation. Celle-ci fait appel à la fonction NACA.create à laquelle elle va donner les arguments bleus pour créer un profil d'aile. NACA.create va créer le fichier NACA_Airfoil.txt. Ce fichier va être utilisé par la fonction Input.setting avec les arguments verts pour appeler Xfoil et stocker les coefficients C_L, C_D, C_M et alpha dans le fichier Result.txt. On utilise ensuite la fonction Output.extract pour récupérer ces coefficients dans DataTreatment. DataTreatment fait ensuite la sélection de la réponse qu'il doit envoyer. Si option=None alors answer permet de directement choisir entre le renvoi de C_L,C_D et C_M. Si option="alpha=2" alors on va demander à Xfoil de faire les calculs pour un alpha précis égal à 2. Si option= "CL=0.5" alors on va demander à Xfoil de faire les calculs pour une valeur précise de C_L. On remarque que l'on a choisi de renvoyer C_D ou $-C_L$. En effet, on utilise une fonction de minimisation à la suite et on cherche bien à maximiser C_L et à minimiser C_D. Si option=average alors on fera une moyenne de tous les coefficients C_L ou C_D selon si on a choisi C_D ou C_L comme valeur de answer.

3 Tests du logiciel Xfoil

Avant d'utiliser Xfoil, il est intéressant de voir si ce simulateur est fiable.

Afin de vérifier la véracité des résultats obtenus par le biais du simulateur Xfoil, nous allons comparer ceux-ci à de vrais tests effectués en soufflerie sur un profil d'aile AG12 dont voici le profil :



La totalité des résultats de soufflerie de ce profil sont fournis en annexe, ici nous nous contenterons de comparer quelques résultats.

Pour tester Xfoil nous avons tracé des courbes de coefficients de portance (à gauche) ainsi que de traînée (à droite) en fonction de alpha pour différents coefficients de Reynolds :

Les courbes en rouge sont celle obtenues par les tests en soufflerie, en bleu par Xfoil.

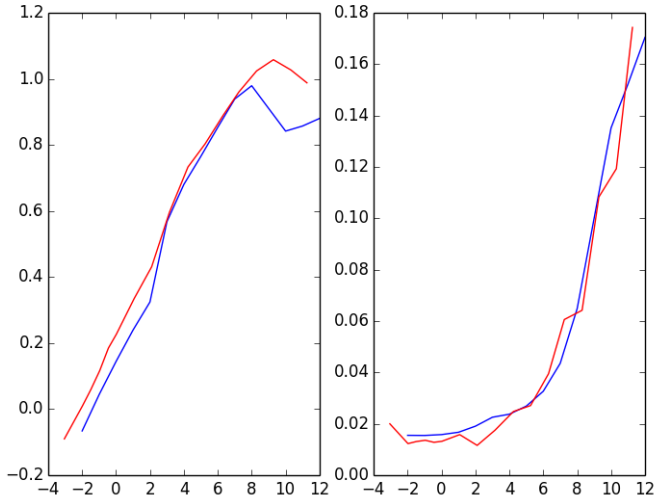


Figure 5: Re=40 000

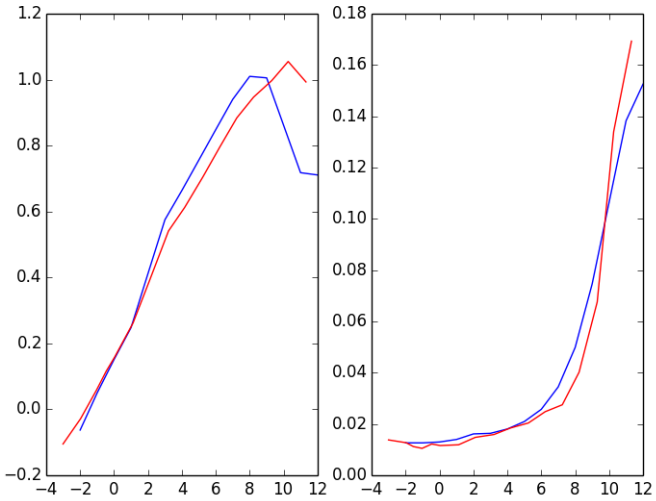


Figure 6: Re=60 000

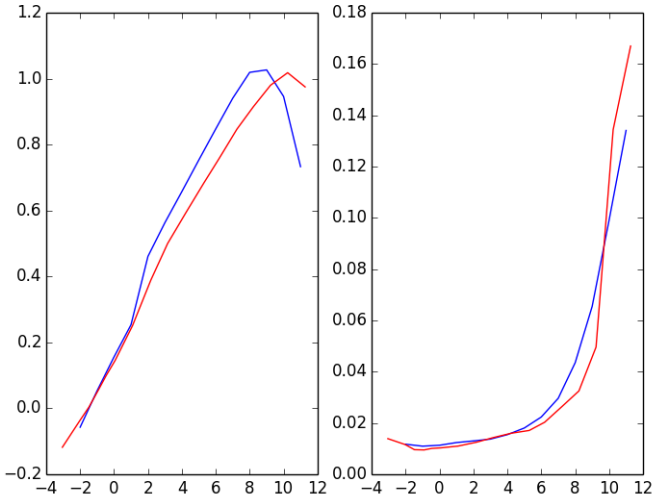


Figure 7: Re=80 000

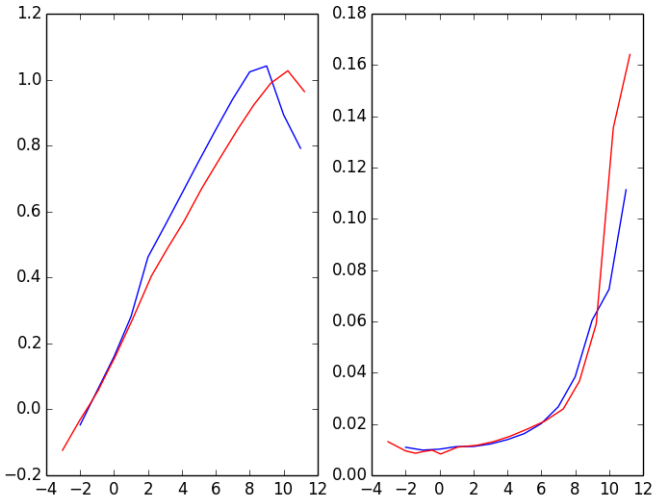


Figure 8: Re=100 000

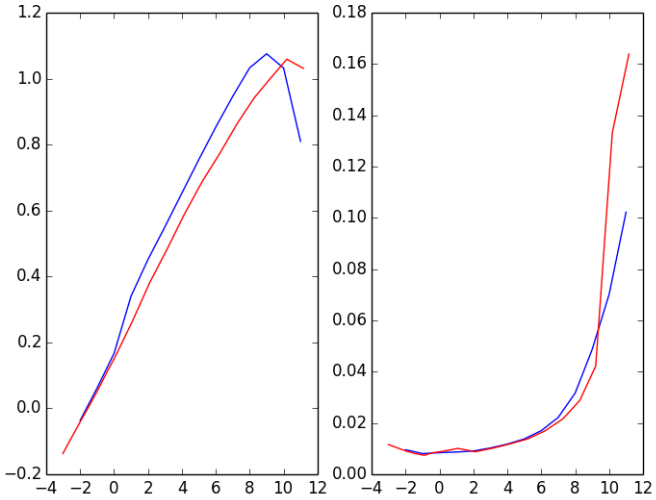


Figure 9: Re=150 000

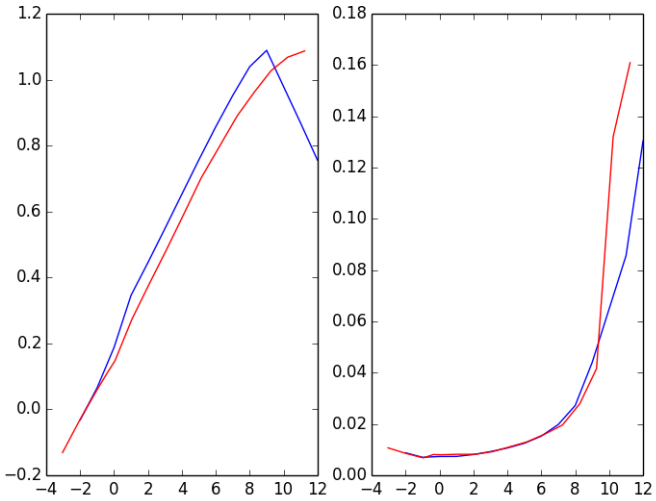


Figure 10: Re=200 000

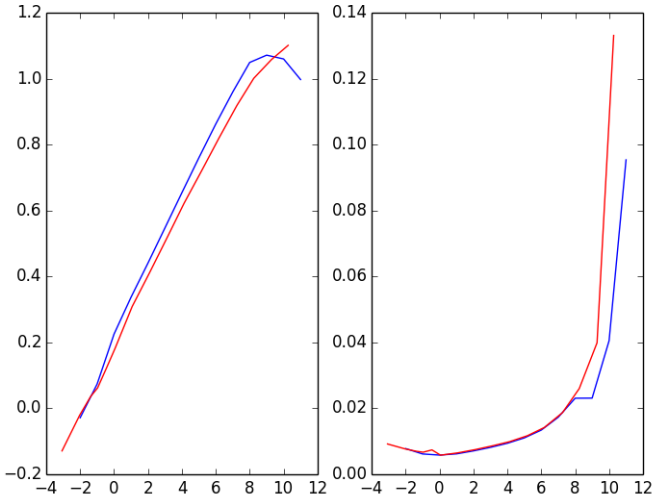


Figure 11: Re=300 000

Du fait des hypothèses simplifiées de Xfoil, l'instant de décrochage est mal simulé, on voit clairement qu'à partir d'un angle d'attaque supérieur à 8° les courbes s'éloigne de manière significative. Ce test nous apprend donc que l'on peut se fier au simulateur pour un angle d'attaque inférieur à 8°.

4 Optimisation du profil

Dans cette partie, on va s'attaquer au cœur du projet : l'optimisation !

Dans ce projet, nous serons amenés à utiliser deux algorithmes d'optimisation : un avec descente de gradient et un autre sans gradient. Celui avec gradient sera un algorithme de BFGS avec approximation du gradient et l'autre sera un algorithme de Nelder-Mead du subplex. Les deux algorithmes sont présentés en annexe.
Rappelons que le but de ce projet est de minimiser la traînée de l'aile, qui sera fait sous contrainte d'un certain coefficient de portance. Mais lequel ?

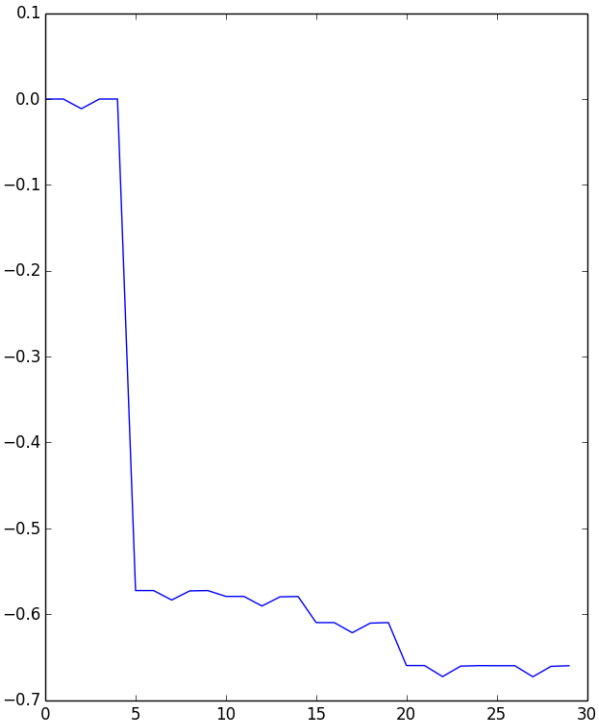
On va donc dans un premier temps chercher à maximiser la portance de l'aile d'avion pour un angle d'attaque égal à 0° afin de savoir quelle portance on pourra exiger lors de la minimisation du coefficient de traînée.

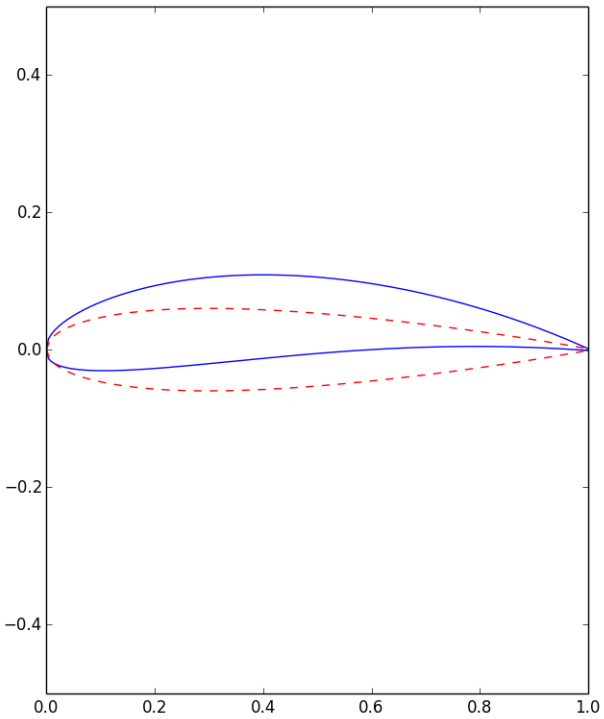
Les résultats d'optimisation seront tous présentés de la même manière : à gauche le suivi de convergence tandis qu'à droite nous aurons les profils de départ en pointillés rouges et d'arrivée en bleu.

4.1 Maximisation de la portance

Avec l'algorithme de BFGS :

L'optimisation aboutit à un coefficient de portance égale à 0.6599 pour un NACA m=0.05, p=0.5 et t=0.127 en partant d'un profil NACA 0012 (soit m=0, p=0 et t=0.12). Ce profil a un CL égale à 0.00663.

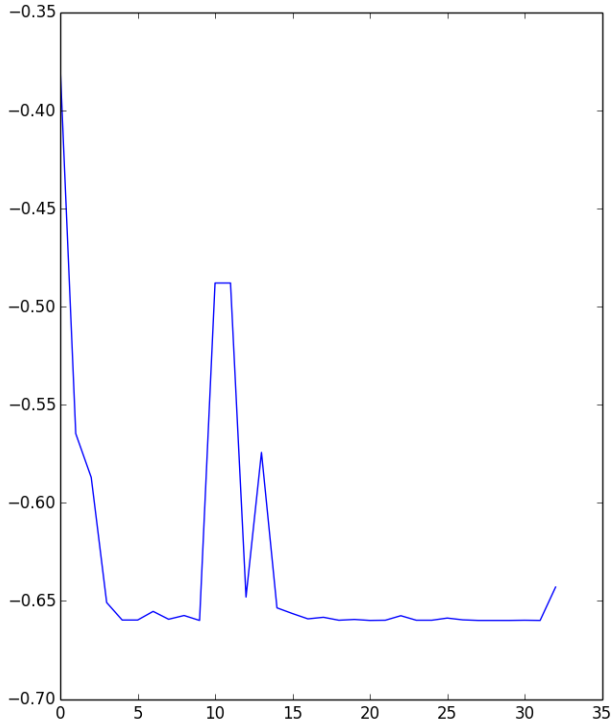


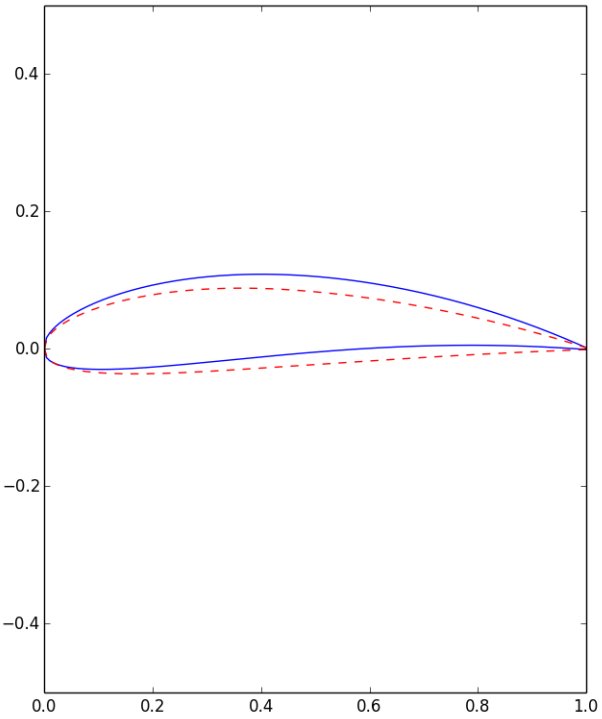


Avec l'algorithme du subplex :

L'optimisation aboutie un coefficient de portance égal à 0.6599 pour un NACA m=0.05, p=0.5 et t=0.127 en partant d'un profil NACA 0012 (soit m=0, p=0 et t=0.12). Ce profil a un CL égale à 0.0065.

Soit les mêmes profils.





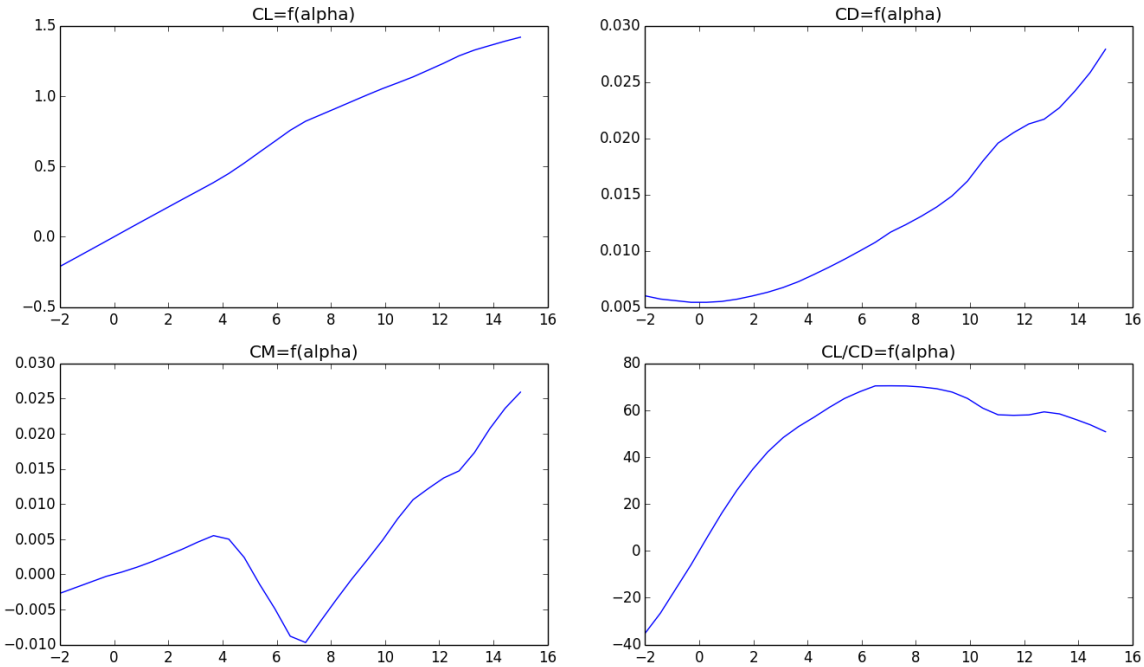
Cette optimisation nous montre que l'on peut demander un coefficient de portance au plus égal à 0.6599. Etant donné qu'il s'agit d'un maximum on ne le choisira pas comme condition. En fait, on prendra CL=0.5 la minimisation de traînée ce qui est suffisant pour que l'avion puisse voler.
Remarquez que l'on peut aisément dépasser 0.0065 mais pour d'autre valeur d'angle d'attaque. Comme nous voulons une aile dont dont les meilleures caractéristiques soient pour un angle d'attaque proche de zéro, il est tout à fait naturel de choisir CL=0.5 et de voir des CL dépasser 0.0065.

4.2 Présentation des profils de départ

Pour la partie qui suit, nous allons prendre deux points de départ afin de voir son influence sur le résultat.

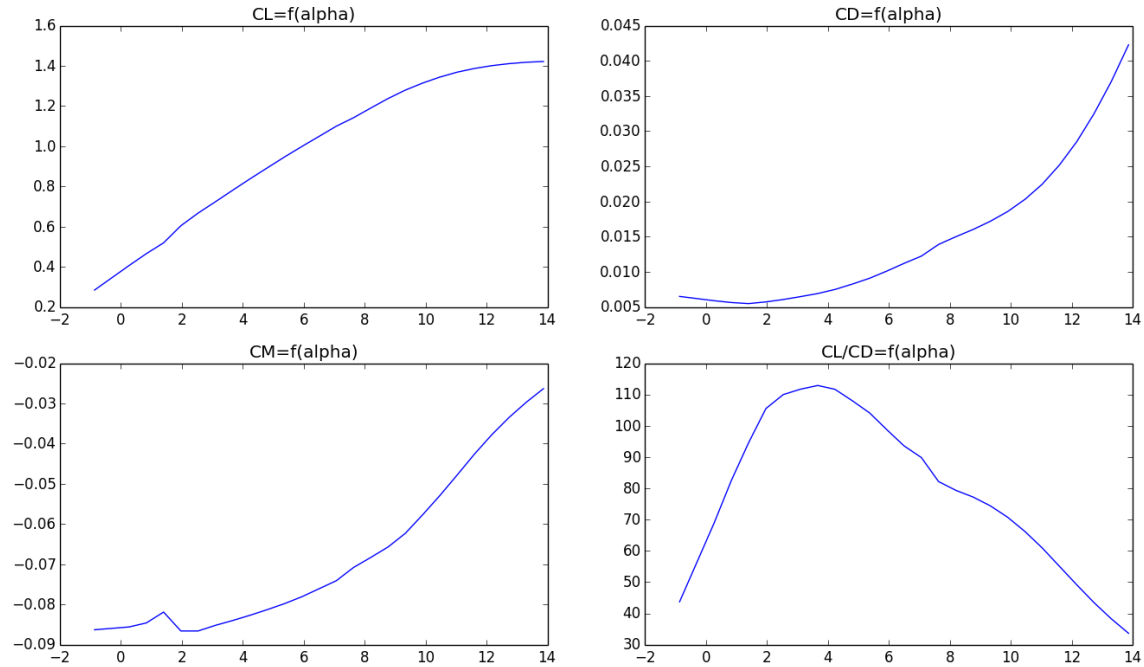
le profil 1:

Le NACA 0012 (m=0, p=0 et t=0.12), un profil symétrique dont voici les courbes caractéristiques:



le profil 2:

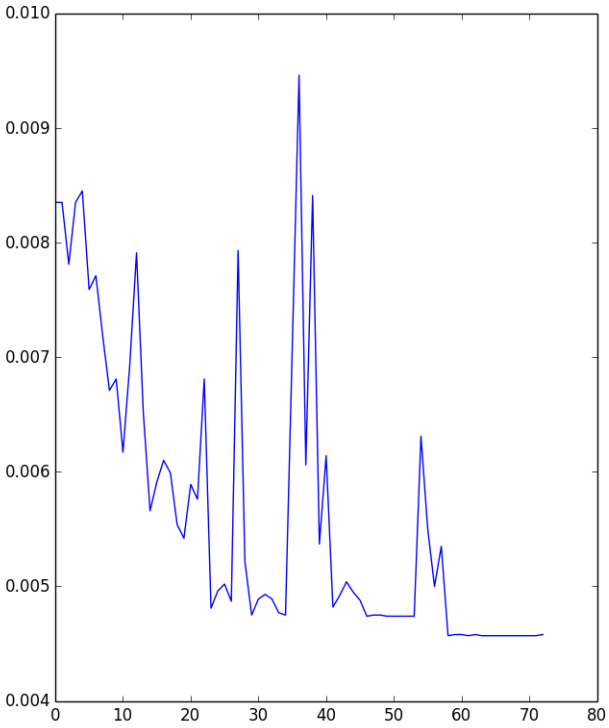
Le NACA m=0.03, p=0.45 et t=0.12 dont voici les courbes caractéristiques:

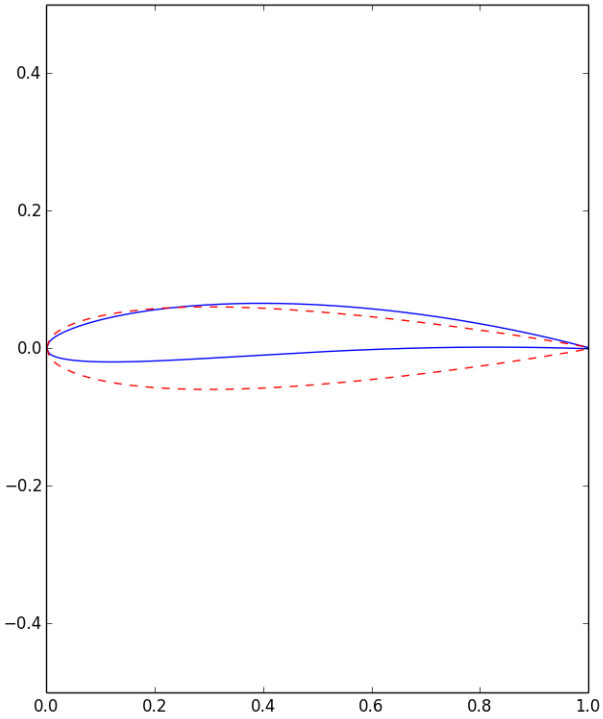


4.3 Minimisation de la traînée (Sous contrainte CL=0.5)

Avec l'algorithme du subplex :

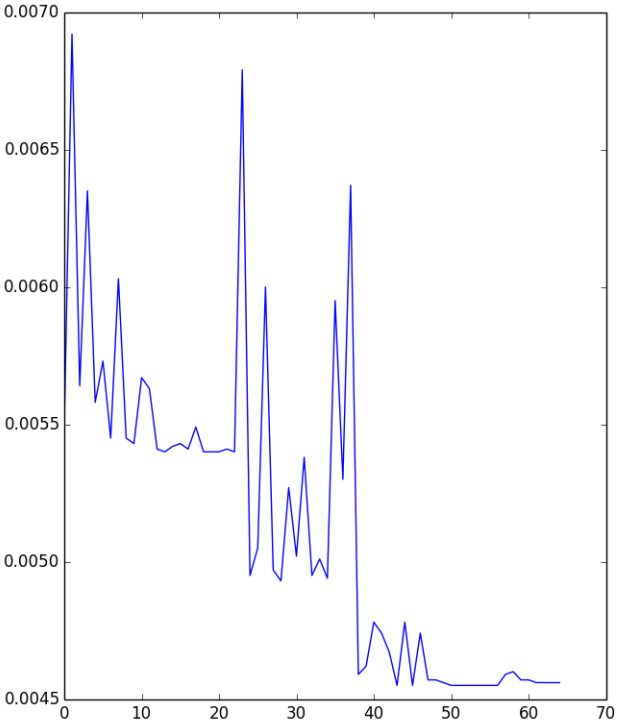
En partant d'un profil NACA 0012, nous arrivons à un profil NACA m=0.0287, p=0.496 et t=0.077 pour un coefficient de traînée de 0.00458.

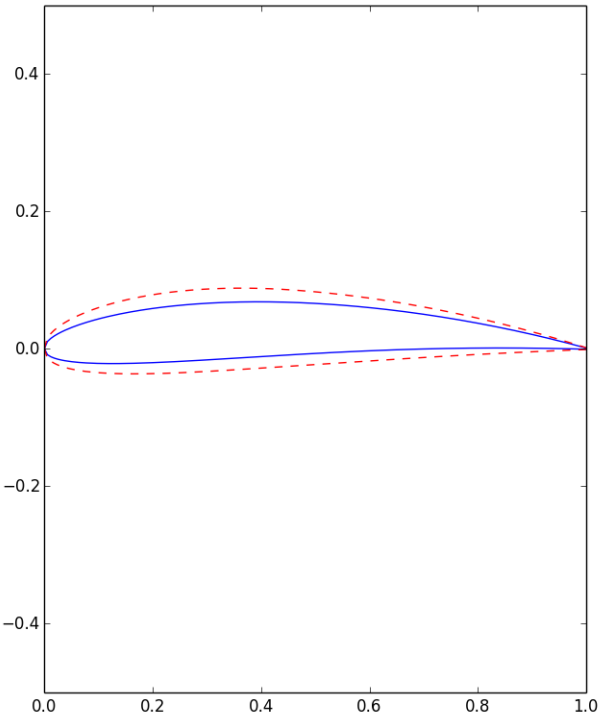




En partant d'un profil NACA m=0.03, p=0.45 et t=0.12, nous arrivons à un profil NACA m=0.0294, p=0.5 et t=0.083 pour un coefficient de traînée de 0.00456.

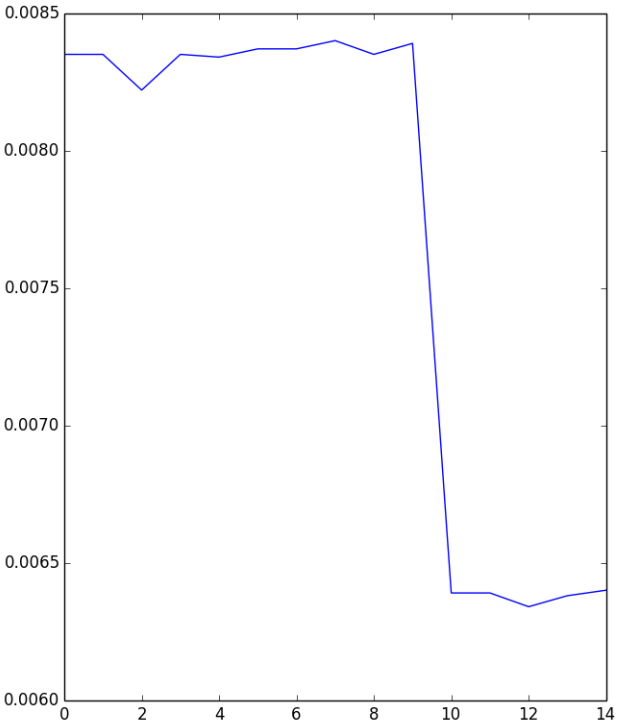
Soient deux profils très proches!

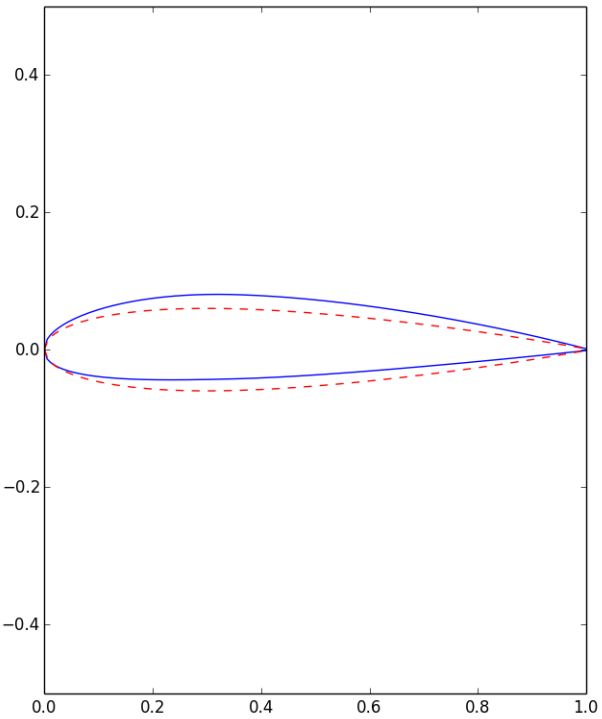




Avec l'algorithme de BFGS :

En partant d'un profil NACA 0012, l'algorithme de gradient converge vers un NACA m=0.019, p=0.35 et t=0.125 pour un CD=0.0064, ce qui est loin d'être le meilleur résultat.

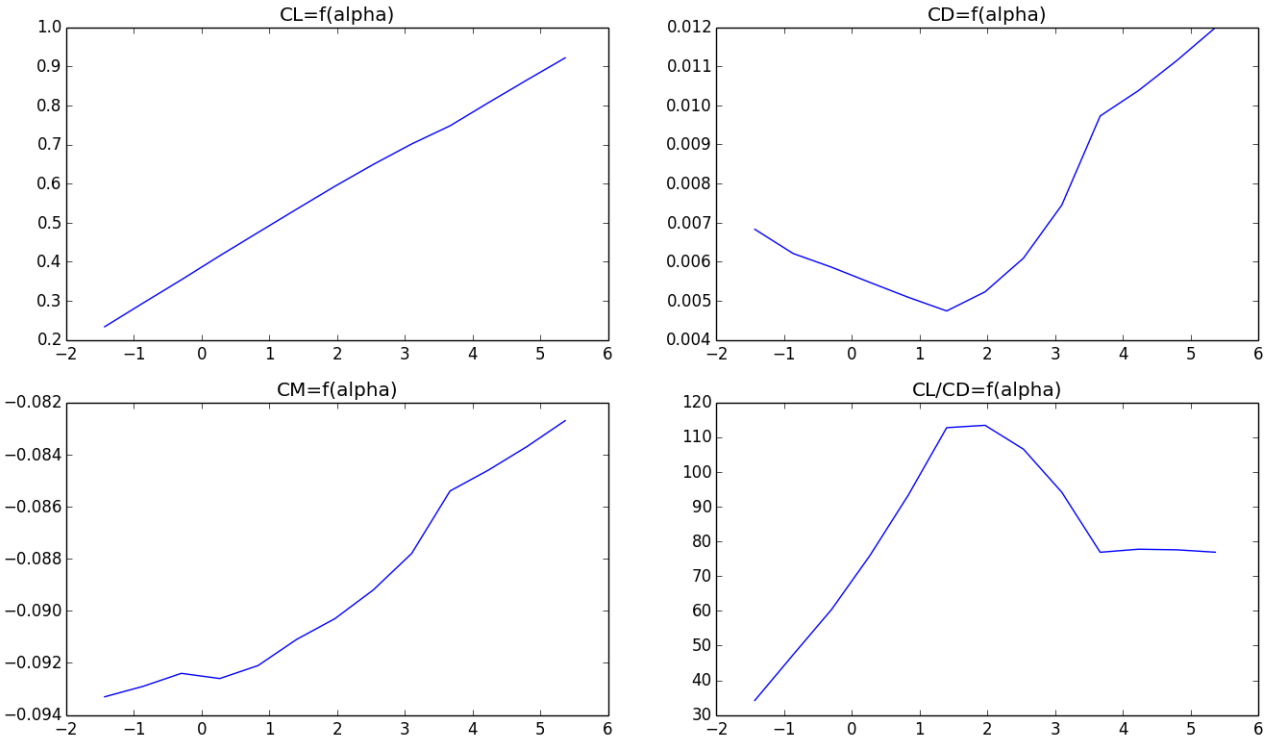




En partant d'un profil NACA m=0.03, p=0.45 et t=0.12, l'algorithme ne converge pas

(cela est dû à la non convergence des calculs sur le simulateur, autant pour l'algorithme du subplex on peut réussir à faire sauter la valeur, autant avec BFGS si on tombe dans une zone de non convergence alors on n'en réchappe pas !)

Voici la présentation des caractéristiques du profil ayant un coefficient de traînée minimum (CD=0.00456) pour CL=0.5 (m=0.0294, p=0.5 et t=0.083):



Les résultats de cette partie nous apprennent donc que si nous voulons trouver la meilleure solution avec BFGS, il faudra étudier d'un peu plus près la fonction DataTreatment. Et qui sait? Peut être qu'un autre minimum va être trouvé...

4.4 Etude de la fonction DataTreatment

Nous allons donc maintenant faire des tracés de la fonction pour différentes valeurs de l'épaisseur d'aile (le paramètre t) allant de 0.05 à 0.20 (se trouveront plus loin dans cette partie que les tracés les plus intéressants).
Les différents tracés représenteront les différentes valeurs de CD pour un CL fixé à 0.5 pour une cambrure m allant de 0 à 0.03 et une cambrure maximale, p, allant de 0 à 0.45.
Pour faire ces tracés, on calculera les CD en certaines coordonnées, on ne gardera que ceux qui convergent dans Xfoil, le reste sera interpolé par des fonctions radiales cubiques.
Les zones sans points bleus sont donc des zones de non convergence de Xfoil.

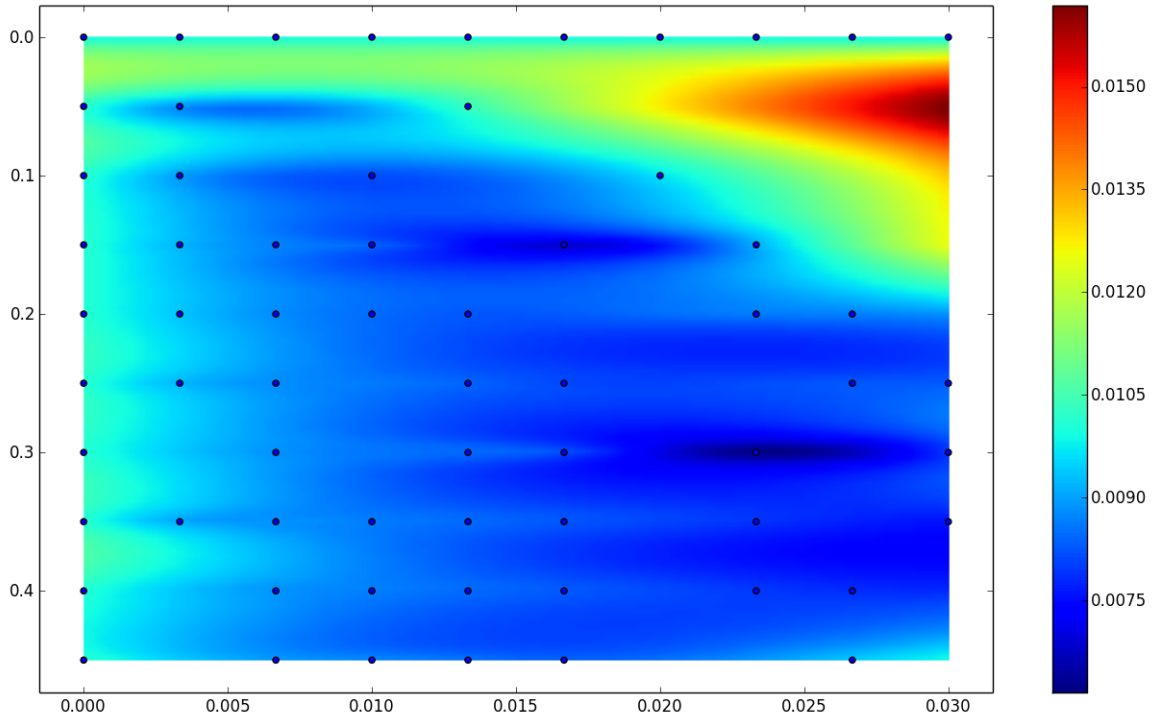


Figure 12: t=0,055

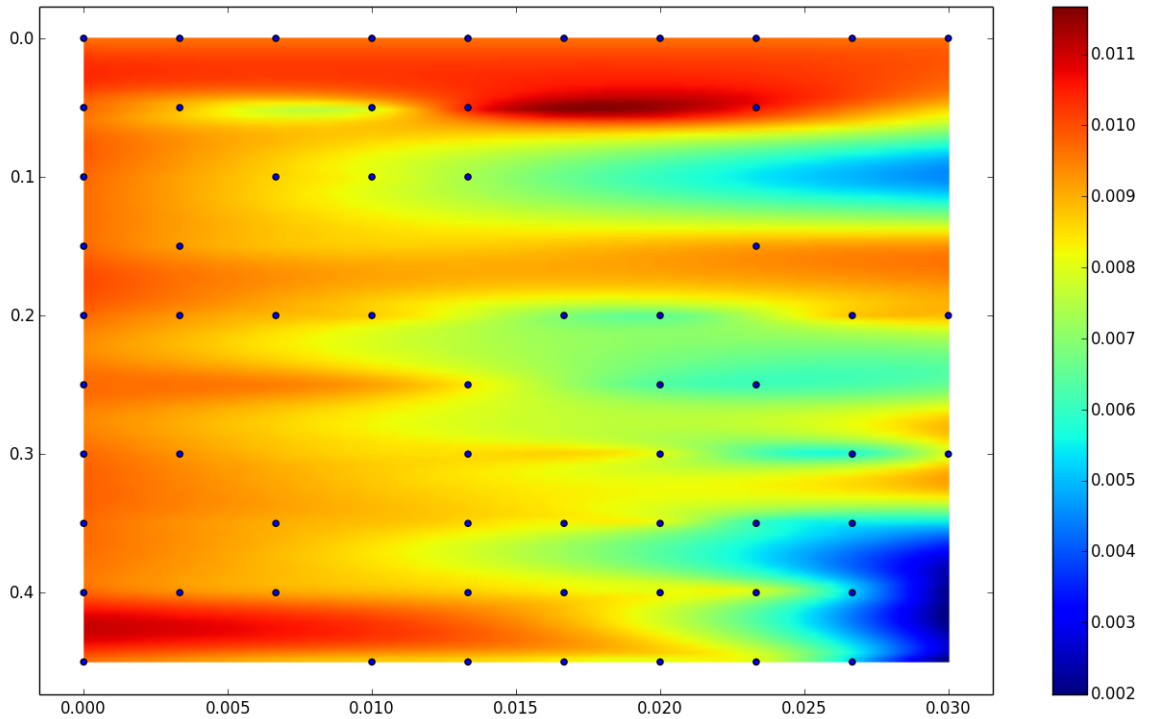


Figure 13: t=0,060

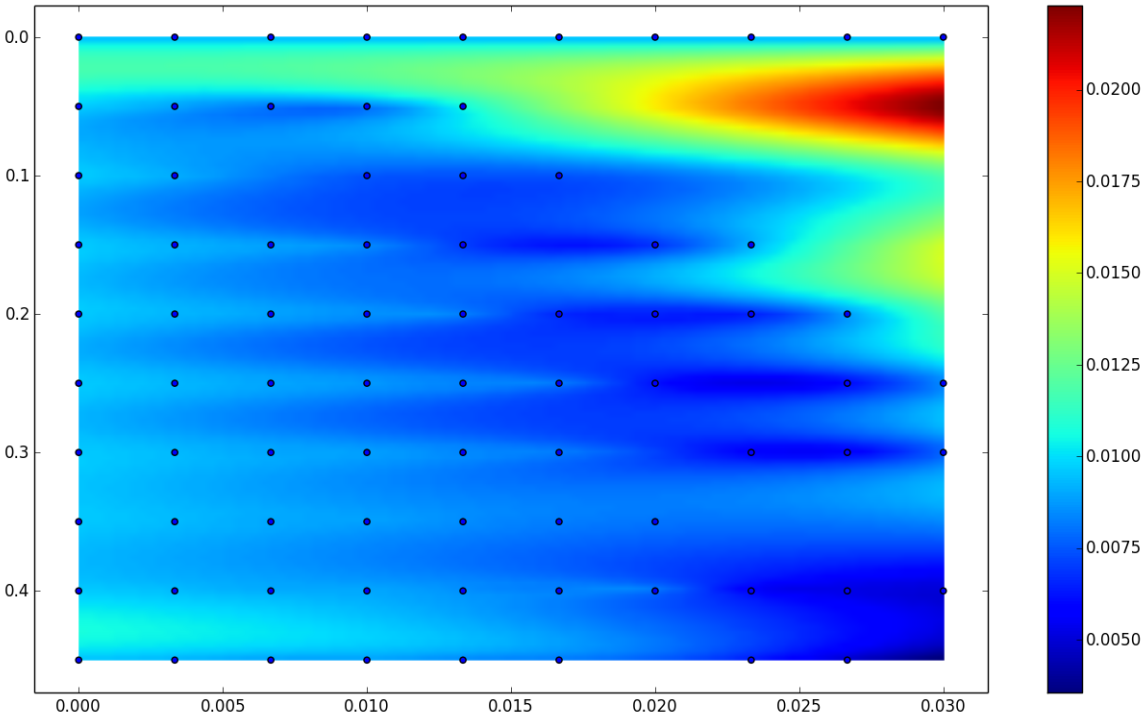


Figure 14: t=0,065

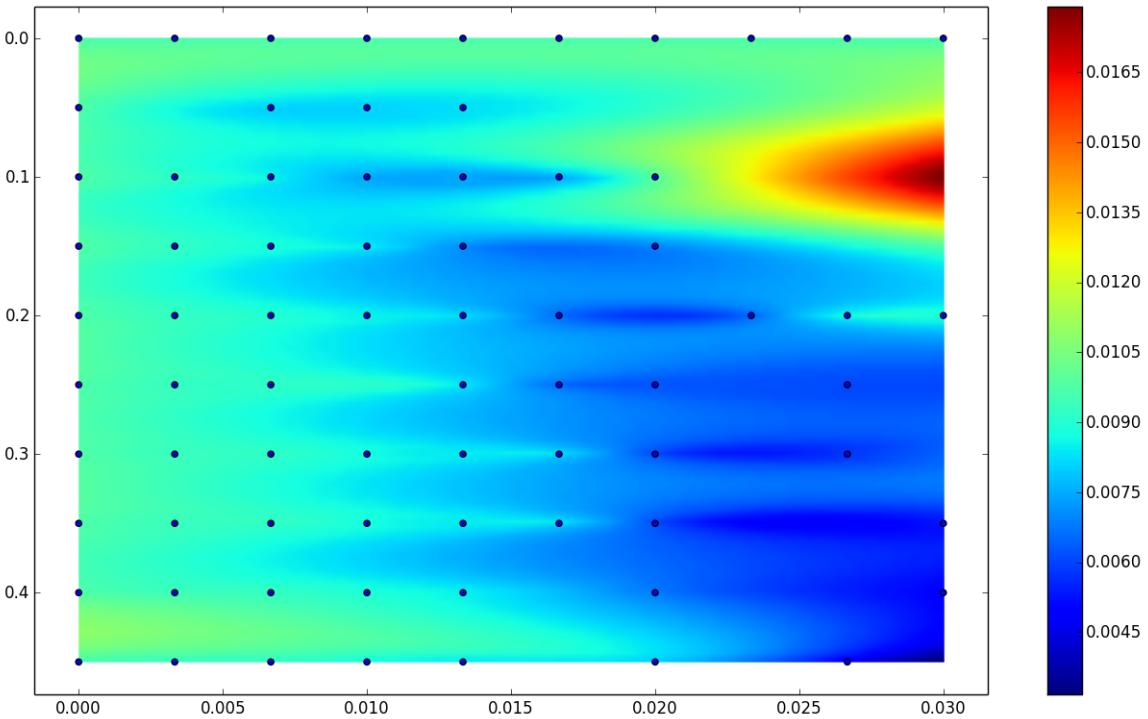


Figure 15: t=0,070

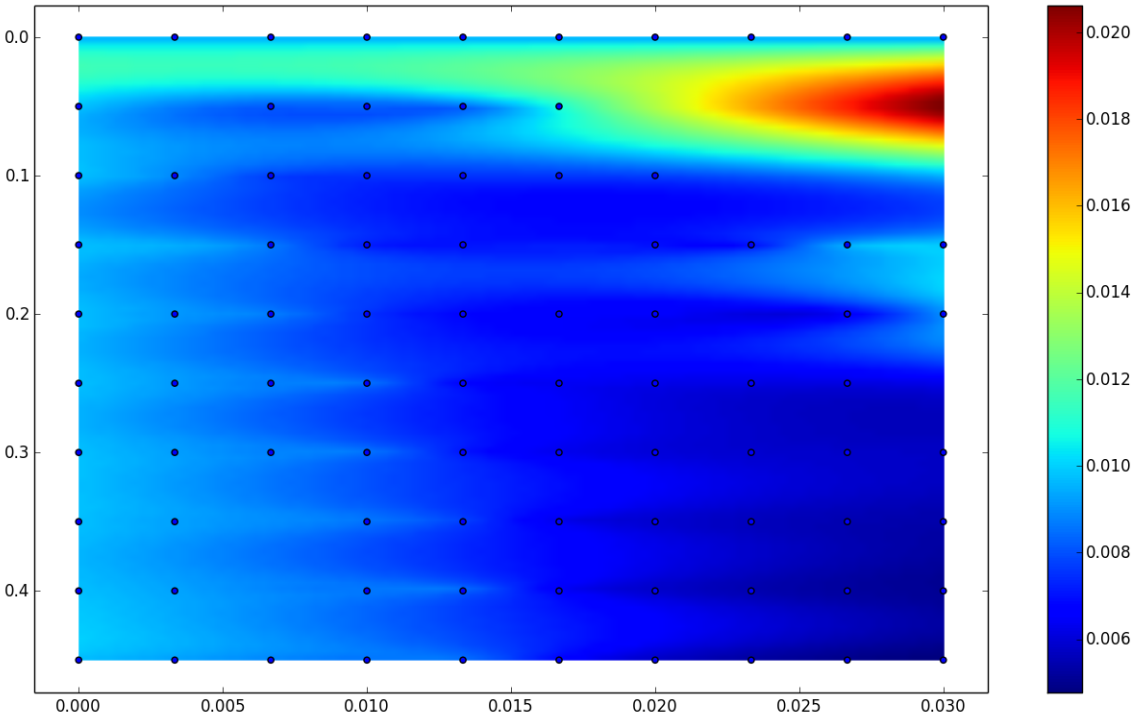


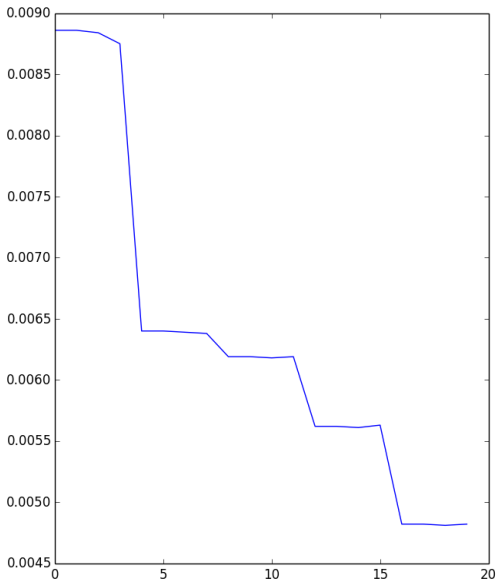
Figure 16: t=0,080

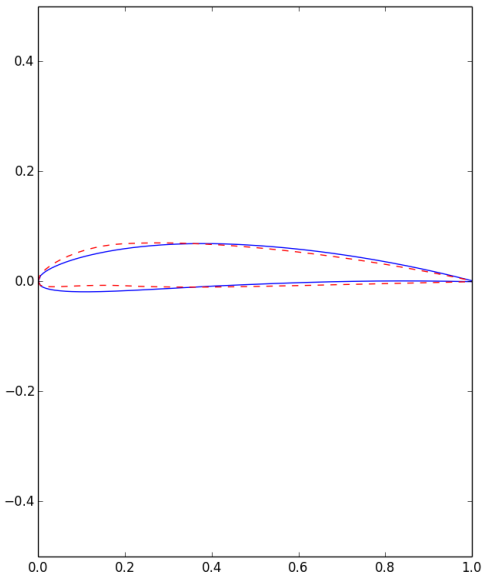
Avec ces différents tracés, nous pouvons remarquer qu'il y a une zone où CD est inférieur à 0.00456 (minimum trouvé dans la partie précédente), et que cette zone correspond à un m environ égal à 0.03, p proche de 0.45 et t compris entre 0.055 et 0.08. D'où l'intérêt de la partie qui suit où nous allons fixer une variable: m=0.03.

4.5 Minimisation de la traînée (Sous contrainte CL=0.5 et m=0.03)

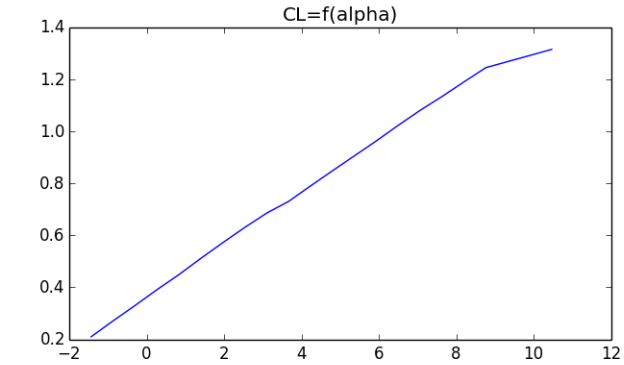
Dans cette partie, nous allons utiliser uniquement l'algorithme de BFGS. En effet, comme nous avons pu le constater, l'algorithme de Nelder-Mead n'est pas affecté par le point de départ.
Au vue des tracés de DataTreatement, afin les zones de non convergence de Xfoil, nous choisirons comme point de départ m=0.03 (contrainte), p=0.2 et t=0.08. Soit un profil dont le coefficient de traînée vaut 0.00886.

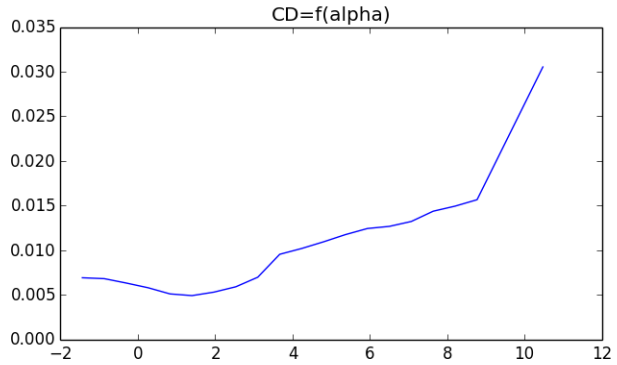
Voici les résultats de cette optimisation:

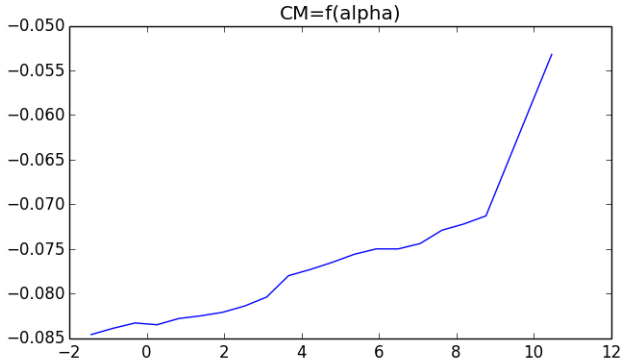


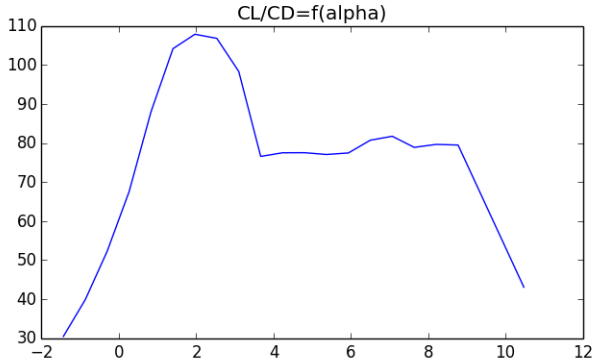


Nous arrivons à un profil NACA m=0.03, p=0.45 et t=0.081 dont le coefficient de frottement pour CL=0.5 égal à CD=0.00482.
Ce n'est pas mieux que lors des parties précédentes, mais au moins nous avons montré qu'en guidant un peu plus l'algorithme de BFGS, nous arrivons à un résultat proche de celui du subplex.
Voici les caractéristiques du profil:









4.6 Optimisation multi-objectifs

Dans les parties précédentes nous nous sommes attachés à trouver le profil minimisant la traînée pour un CL fixé à 0.5. Mais pourquoi s'arrêter là ? Un avion n'est jamais confronté à CL fixé à 0.5 !
En effet, quand nous comparons les caractéristiques des deux profils de départ avec le profil optimisé pour CL=0.5, voici ce que nous obtenons:

Les CD moyens sur une plage de CL compris entre 0.5 et 0,7 sont:

- profil 1: 0.009409 (0.00835 à CL=0.5)
- profil 2: 0.005884 (0.00549 à CL=0.5)
- profil optimisé: 0.00569 (0.00456 à CL=0.5)

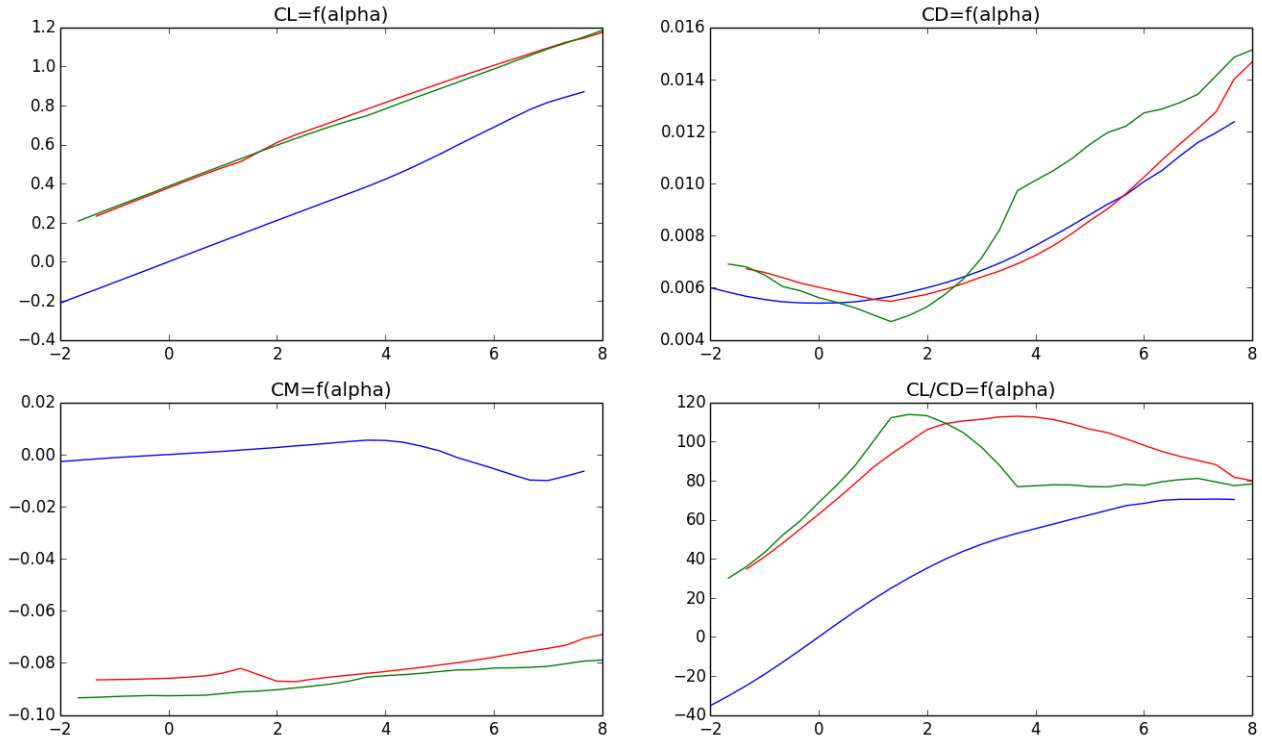
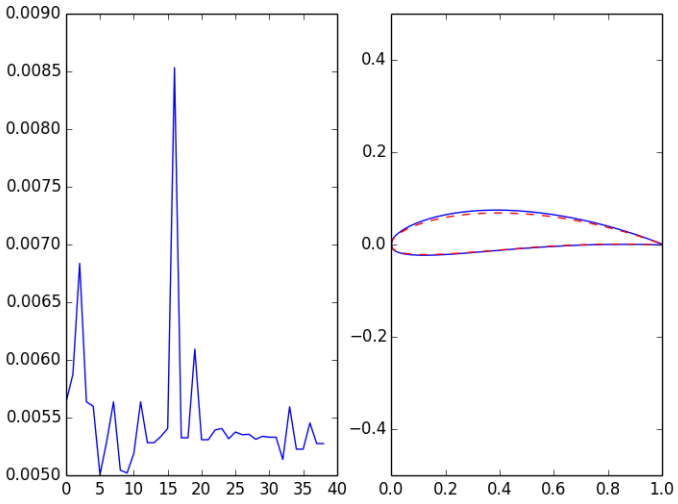


Figure 17: bleu:profil 1, rouge:profil 2, vert: profil optimisé

Certes, notre profil optimisé est encore meilleur pour la valeur moyenne de CD.
Cependant, au vu de la courbe de CD(α) (pointe), on peut se douter qu'il doit y avoir des profils plus performants sur une plage de alpha.
C'est pourquoi nous allons faire de l'optimisation multi-objectifs: il s'agit de faire une moyenne (normalement pondérée, mais pour cela nous devrions avoir des informations sur les temps de vol moyen pour les différents CL) de coefficients de traînée sur une plage donnée de CL (de 0.5 à 0.7).

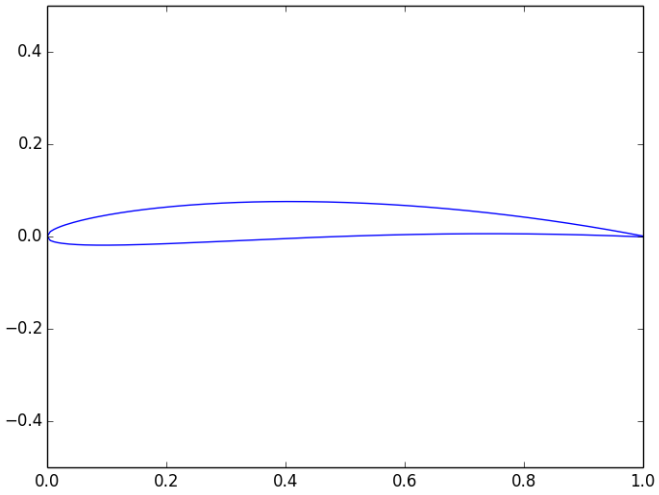
Avec l'algorithme de BFGS, nous n'aboutissons à rien.

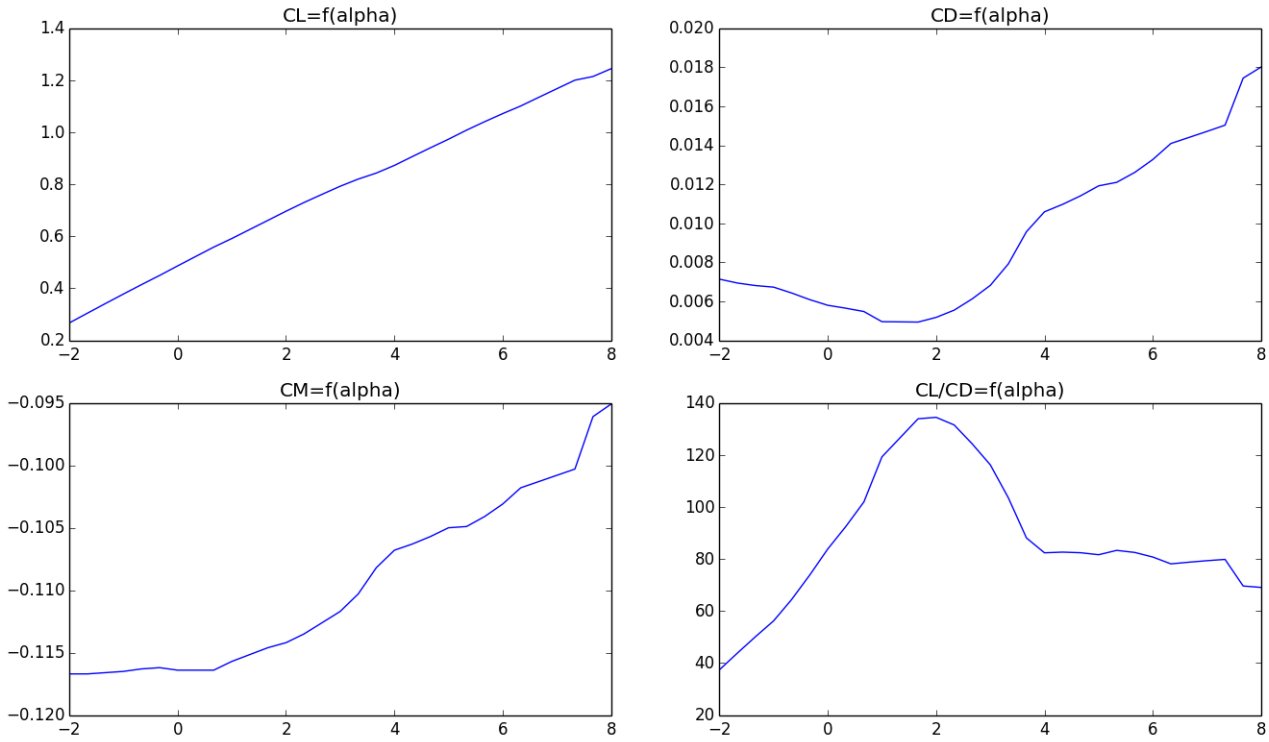
Par contre, avec celui du subplex, en partant du profil optimisé, voici ce que obtenons:



L'algorithme converge vers un NACA m=0.0318863966972 p=0.479883521867 t=0.0900768126702 avec CD moyen égal à 0.005275.

Cependant on peut voir qu'il y a des itérations plus performantes, dont celle ci: CD_{moy}=0.005005, m=0.037125, p=0.5 et t=0.083





Remarquez que nous avons pris une plage de CL plutôt restreinte, ceci afin de permettre une (pseudo) convergence d'un des algorithmes.
Ceci explique pourquoi nous n'irons pas plus loin dans cette partie.

5 Conclusion

Nous avons débuté ce projet par créer la fonction DataTreatment qui permet de calculer les coefficients C_L et C_D en fonction d'un profil NACA donné. Nous avons ensuite intégré la fonction DataTreatment dans un procédé d'optimisation en utilisant deux types d'algorithmes: méthode avec descente de gradient et méthode du subplex.

Nous cherchions à obtenir une consommation en carburant la plus faible possible lors du trajet d'un avion. Cela revient à chercher à minimiser la traînée en conservant une portance suffisante pour faire voler l'avion. Un des aspects importants de ce projet est que nous avons avancé progressivement afin obtenir un profil aux propriétés satisfaisantes. Nous avons rapidement vu que la méthode avec descente de gradient présente plusieurs problèmes du fait de sa sensibilité au point initial. La méthode du subplex donne des résultats satisfaisants mais on se rend compte que de meilleurs résultats sont possibles en traçant une représentation des valeurs de DataTreatment. Un moyen simple pour obtenir de bons résultats avec BFGS est de fixer certains paramètres que nous savons corrects pour optimiser sur les variables restantes. Cependant les profils trouvés ne seront valables que sous certaines hypothèses, on ne peut pas vraiment s'arrêter sur un profil en particulier car cela dépend avant tout des critères d'optimisation.

Le profil sortant de la partie 4.3 (NACA m=0.019, p=0.35 et t=0.125 pour un CD=0.0064) est celui, qui selon nos premiers critères, est le meilleur en terme de coefficient de traînée. Ce profil nécessite néanmoins d'avoir un CL constant égal à 0.5 ce qui correspond à la portance que l'on doit satisfaire lorsqu'un avion est un croisière.

Si nous voulons étudier le meilleur profil pour un plan de vol complet, nous devons considérer le fait qu'à certains moments nous allons besoin d'un CL plus élevé notamment au décollage où la vitesse de l'avion est plus faible ce qui doit être compensé par un CL plus fort. La méthode d'usage pour résoudre ce problème est l'optimisation multi-objectifs. Notre courte optimisation multi-objectif nous révèle que sur une plage de CL (assez courte) nous pouvons trouver mieux (NACA m=0.037125, p=0.5 et t=0.083, CD_{moy}=0.005005). Et nous aurions sûrement trouvé un autre résultat si nous avions pu augmenter la plage de CL.

Si nous avions à continuer l'étude, nous aurions deux pistes à creuser:

- La première serait d'améliorer la résolution multi-objectif.
- La deuxième serait de se placer dans un cas 3D en utilisant notre méthode 2D: créer un modèle 3D à partir de profils NACA (comme cela a pu être fait sur la page de garde) et considérer des tranches d'aile comme des profil 2D afin d'obtenir un résultat grâce aux outils décrits et utilisés dans ce projet.

6 Annexes

6.1 Implémentation des algorithmes d'optimisation

On utilise des fonctions d'optimisation afin de déterminer le meilleur profil NACA en fonction de critères sur les coefficients de portance et de trainée. Nous allons utiliser l'algorithme BFGS de la librairie scipy.optimize et une variante de l'algorithme du simplexe de Nelder-Mead de la librairie nlopt.

6.1.1 Algorithme BFGS

Cet algorithme permet de résoudre un problème d'optimisation non linéaire sans contraintes. Il fait partie de la famille des algorithmes à direction de descente. Il se base sur la méthode de Newton mais utilise une approximation de la Hessienne. L'appel de cet algorithme dans notre programme se fait avec la fonction fmin_l_bfgs_b. Ses arguments sont :

- DataTreament : la fonction à minimiser
- x0=x : le point de départ pour chercher le minimiseur
- fprime=none : pour cet algorithme, il faut donner un gradient, ici on laisse fmin_bfgs calculer une approximation
- args=arg : les arguments que l'on passe à la fonction DataTreatment
- approx_grad=True : fmin_bfgs calcule une approximation du gradient
- bounds=b : les limites à l'intérieur desquelles les arguments m,p et t du profil NACA doivent se trouver
- m=10 : un seuil qui limite la taille de la Hessienne à stocker
- pgtol : définit un critère d'arrêt de l'algorithme basée sur le gradient (si une coordonnée du gradient est inférieur à ce nombre on stoppe l'algorithme)
- factr=1000000.0 : définit un critère d'arrêt de l'algorithme (la différence relative de deux itérations successives est inférieure à la précision machine fois ce nombre)
- epsilon=1e-03 : taille du pas pour calculer l'approximation du gradient

6.1.2 Algorithme de Nelder-Mead

L'algorithme du simplexe de Nelder-Mead a l'avantage par rapport à la méthode précédente de ne pas nécessiter le gradient de la fonction à minimiser. Ce type d'algorithme se base sur un simplexe, c'est-à-dire une enveloppe convexe formé par des points non alignés. Dans notre cas, on utilise l'algorithme pour un problème de dimension 3. Le simplexe est donc un ensemble de quatre points de R^3. Le but de cet algorithme est d'effectuer des expansions ou des contractions du simplexe en fonction de la topologie locale de la fonction à minimiser. L'algorithme va chercher en quel point parmi les quatre la fonction est la plus élevée pour le remplacer par un nouveau point. Le nouveau point sera calculé en fonction des trois restants. On réitère ce processus jusqu'à obtenir une convergence. D'une manière générale, l'algorithme de Nelder-Mead fonctionne bien d'une manière générale . Il existe cependant des fonctions pour lesquelles cet algorithme échoue à partir de la dimension 2. Dans notre cas, la librairie nlopt utilise l'algorithme du "subplex" qui est une variante de l'algorithme de Nelder-Mead, plus robuste et plus efficace que ce dernier. L'appel de cet algorithme dans notre programme se fait de la manière suivante :

opt = nlopt.opt(nlopt.LN_SBPLX,3) % Sélection de l'algorithme du subplex
opt.set_lower_bounds([-0.05,0.35,0.05]) % Sélection des bornes dans lesquelles doivent être les résultats
opt.set_upper_bounds([0.05,0.5,0.2])
opt.set_min_objective(myfunc) % On cherche le minimum de la fonction

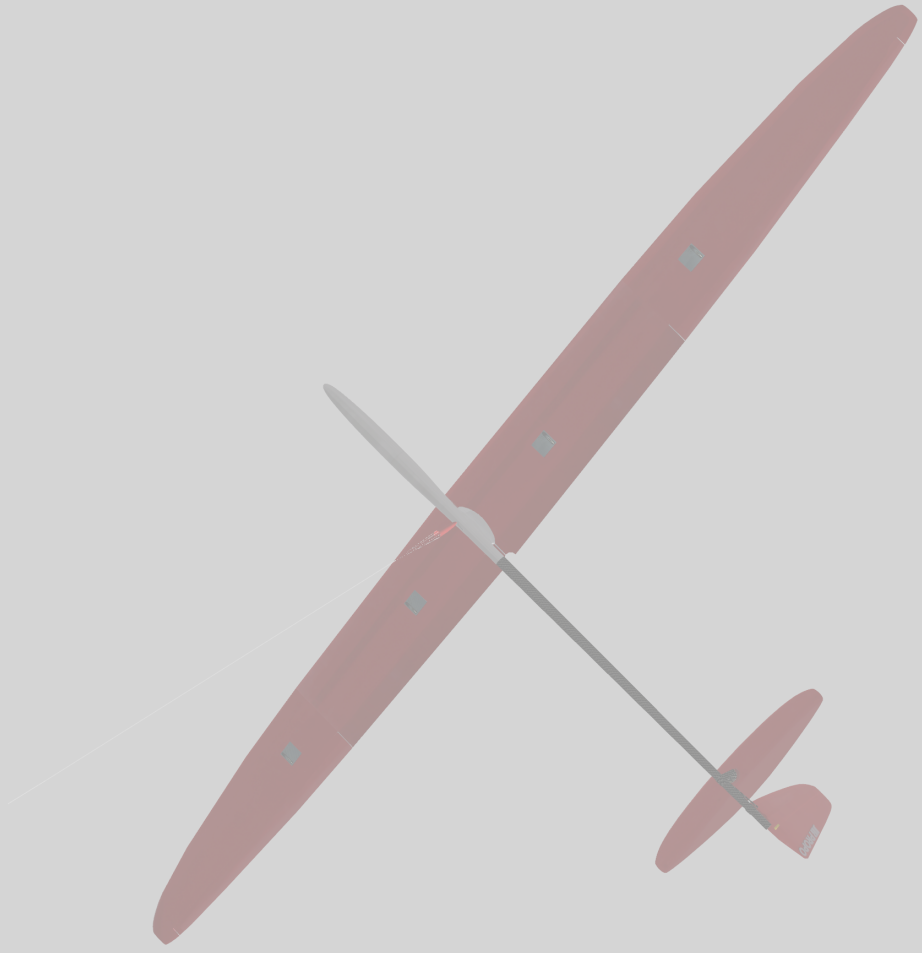
opt.set_xtol_rel(5e-2) % Définition de la tolérance
x0=array([0.03, 0.45,0.12]) % Point initial

x = opt.optimize(x0) % On lance l'algorithme choisi avec le point initial
minf = opt.last_optimum_value() % On récupère le minimum

6.2 Données expérimentales de soufflerie

Summary of Low-Speed Airfoil Data

Gregory A. Williamson, Bryan D. McGranahan, Benjamin A. Broughton, Robert W. Deters,
John B. Brandt, and Michael S. Selig



Volume 5

AG12
True

x/c	y/c
1.000000	0.000471
0.994142	0.001042
0.982204	0.002222
0.968696	0.003522
0.954910	0.004823
0.941098	0.006097
0.927274	0.007345
0.913441	0.008575
0.899593	0.009787
0.885734	0.010985
0.871861	0.012172
0.857979	0.013351
0.844090	0.014522
0.830201	0.015686
0.816312	0.016845
0.802424	0.017996
0.788539	0.019141
0.774659	0.020278
0.760778	0.021407
0.746901	0.022526
0.733027	0.023634
0.719152	0.024731
0.705280	0.025817
0.691409	0.026889
0.677538	0.027948
0.663670	0.028994
0.649803	0.030024
0.635935	0.031038
0.622068	0.032036
0.608205	0.033016
0.594342	0.033978
0.580478	0.034920
0.566619	0.035841
0.552761	0.036741
0.538905	0.037617
0.525049	0.038470
0.511197	0.039296
0.497347	0.040096
0.483500	0.040866
0.469656	0.041607
0.455816	0.042313
0.441978	0.042986
0.428143	0.043622
0.414316	0.044218
0.400492	0.044774
0.386673	0.045285
0.372861	0.045751
0.359053	0.046165
0.345250	0.046528
0.331461	0.046836
0.317683	0.047081
0.303912	0.047262
0.290156	0.047375
0.276418	0.047413
0.262691	0.047371
0.248985	0.047243
0.235302	0.047024
0.221638	0.046704
0.208004	0.046279
0.194402	0.045737
0.180830	0.045066
0.167304	0.044260
0.153819	0.043301
0.140392	0.042180
0.127049	0.040877
0.113793	0.039371
0.100663	0.037640
0.087677	0.035653
0.074897	0.033377
0.062374	0.030770
0.050223	0.027789
0.038618	0.024400
0.027917	0.020611
0.018756	0.016601
0.011861	0.012819
0.007244	0.009642
0.004260	0.007074
0.002309	0.004947
0.001028	0.003119
0.000261	0.001485
0.000000	-0.000011
0.000299	-0.001481
0.001263	-0.002904
0.002871	-0.004229
0.005195	-0.005565
0.008563	-0.007032
0.013654	-0.008722
0.021338	-0.010619
0.031664	-0.012466
0.043571	-0.013982
0.056254	-0.015147
0.069286	-0.016003
0.082543	-0.016609
0.095952	-0.017016
0.109468	-0.017263
0.123058	-0.017375
0.136704	-0.017378
0.150385	-0.017283
0.164098	-0.017105
0.177834	-0.016851
0.191595	-0.016531
0.205382	-0.016156
0.219178	-0.015730
0.232992	-0.015261
0.246823	-0.014753
0.260663	-0.014213
0.274519	-0.013643
0.288396	-0.013052
0.302276	-0.012444
0.316162	-0.011821
0.330060	-0.011187
0.343964	-0.010546
0.357874	-0.009899
0.371793	-0.009250
0.385719	-0.008601
0.399644	-0.007955
0.413564	-0.007312
0.427475	-0.006678
0.441384	-0.006050
0.455289	-0.005434
0.469185	-0.004829
0.483074	-0.004240
0.496967	-0.003667
0.510856	-0.003112
0.524739	-0.002575
0.538618	-0.002060
0.552501	-0.001566
0.566379	-0.001094
0.580251	-0.000648
0.594125	-0.000227
0.608000	0.000167
0.621871	0.000536
0.635737	0.000876
0.649609	0.001189
0.663478	0.001474
0.677342	0.001730
0.691207	0.001956
0.705074	0.002153
0.718938	0.002321
0.732801	0.002458
0.746669	0.002565
0.760534	0.002643
0.774396	0.002689
0.788262	0.002707
0.802128	0.002695
0.815988	0.002653
0.829854	0.002581
0.843721	0.002481
0.857585	0.002351
0.871458	0.002194
0.885322	0.002011
0.899184	0.001798
0.913057	0.001557
0.926935	0.001293
0.940803	0.001006

AG12
Fig. 4.3

Run: bb05707_interp
Re = 39617.3

α	C_l	C_d
-3.05	-0.090	0.0200
-1.98	0.010	0.0123
-1.49	0.059	0.0131
-0.96	0.117	0.0136
-0.45	0.185	0.0128
0.02	0.227	0.0132
1.07	0.335	0.0158
2.10	0.431	0.0116
3.15	0.596	0.0175
4.24	0.733	0.0248
5.25	0.803	0.0271
6.31	0.889	0.0395
7.24	0.960	0.0606
8.29	1.024	0.0642
9.28	1.058	0.1083
10.31	1.027	0.1193
11.26	0.988	0.1743

Run: bb05637_interp
Re = 59971.8

α	C_l	C_d
-3.01	-0.105	0.0138
-1.97	-0.028	0.0127
-1.54	0.012	0.0112
-1.03	0.059	0.0105
-0.46	0.116	0.0122
0.04	0.159	0.0116
1.12	0.262	0.0119
2.11	0.393	0.0148
3.21	0.541	0.0159
4.16	0.612	0.0184
5.24	0.705	0.0204
6.23	0.795	0.0248
7.24	0.884	0.0275
8.23	0.947	0.0402
9.30	0.997	0.0676
10.27	1.055	0.1340
11.32	0.993	0.1693

Run: ts05641_interp
Re = 79922.2

α	C_l	C_d
-3.05	-0.118	0.0139
-2.03	-0.038	0.0116
-1.48	0.004	0.0096
-0.91	0.057	0.0095
-0.46	0.100	0.0101
0.06	0.145	0.0103
1.09	0.251	0.0110
2.17	0.389	0.0125
3.16	0.500	0.0143
4.22	0.592	0.0160
5.29	0.683	0.0171
6.20	0.758	0.0203
7.25	0.847	0.0266
8.22	0.915	0.0325
9.23	0.980	0.0496
10.24	1.018	0.1344
11.27	0.975	0.1670

Run: bb05639_interp
Re = 100021.3

α	C_l	C_d
-3.04	-0.124	0.0131
-2.00	-0.031	0.0095
-1.42	0.016	0.0086
-0.92	0.059	0.0093
-0.43	0.110	0.0098
0.06	0.159	0.0083
1.07	0.271	0.0110
2.21	0.405	0.0117
3.17	0.490	0.0131
4.11	0.569	0.0151
5.17	0.670	0.0180
6.20	0.758	0.0211
7.30	0.850	0.0259
8.25	0.924	0.0367
9.24	0.989	0.0590
10.25	1.027	0.1355
11.23	0.964	0.1641

Run: jb05643_interp
Re = 149943.1

α	C_l	C_d
-3.02	-0.137	0.0116
-1.99	-0.040	0.0091
-1.45	0.009	0.0081
-0.90	0.061	0.0074
-0.43	0.109	0.0084
0.06	0.156	0.0088
1.08	0.264	0.0101
2.11	0.382	0.0088
3.14	0.484	0.0102
4.12	0.586	0.0118
5.20	0.688	0.0138
6.21	0.771	0.0169
7.29	0.866	0.0217
8.27	0.942	0.0288
9.21	1.000	0.0422
10.19	1.059	0.1332
11.17	1.031	0.1639

Run: jb05645_interp
Re = 200448.7

α	C_l	C_d
-3.04	-0.131	0.0107
-2.02	-0.032	0.0086
-1.44	0.020	0.0076
-0.96	0.063	0.0068
-0.36	0.114	0.0081
0.05	0.147	0.0080
1.06	0.274	0.0082
2.12	0.385	0.0082
3.11	0.486	0.0092
4.14	0.595	0.0111
5.14	0.702	0.0130
6.19	0.795	0.0160
7.25	0.889	0.0196
8.27	0.962	0.0279
9.26	1.027	0.0416
10.24	1.068	0.1320
11.24	1.087	0.1609

Run: jb05647_interp
Re = 300060.2

α	C_l	C_d
-3.07	-0.129	0.0092
-2.03	-0.021	0.0077
-1.38	0.036	0.0070
-0.97	0.062	0.0067
-0.45	0.122	0.0074
0.06	0.182	0.0058
1.07	0.309	0.0065
2.11	0.413	0.0075
3.17	0.522	0.0087
4.13	0.622	0.0099
5.18	0.722	0.0117
6.18	0.819	0.0142
7.25	0.919	0.0187
8.24	1.001	0.0260
9.29	1.058	0.0399
10.27	1.101	0.1331

AG16
Fig. 4.7

Run: bb05695_interp
Re = 40171.5

α	C_l	C_d
-3.10	-0.111	0.0161
-2.05	0.007	0.0136

7 Références bibliographiques

NACA Airfoil. [**en ligne**]. Disponible sur : http://fr.wikipedia.org/wiki/Profil_NACA

Mark DRELA, Harold YOUNGREN. *XFoil Subsonic Airfoil Development System*. [**en ligne**]. Disponible sur : http://web.mit.edu/drela/Public/web/xfoil/xfoil_doc.txt

John H. MATHEWS. *Nelder-Mead Method*. [**en ligne**]. Disponible sur : http://mathfaculty.fullerton.edu/mathews/n2003/neldermead/NelderMeadProof.pdf

David M. COOKE. *Optimization(scipy.optimize)*. [**en ligne**]. Disponible sur : http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_l_bfgs_b.htmlscipy.optimize.fmin_l_bfgs_b

Ladislav LUKSAN. *NLopt*. [**en ligne**]. Disponible sur : http://ab-initio.mit.edu/wiki/index.php/NLopt