VERS UNE AVIATION DURABLE

Rapport Final

Auteurs :

Lefebvre Inès
Neveux Baptiste
Larrouturou David
Leene Jules

27 janvier 2026

Table des matières

1 Définition du projet et méthodologie

1.1 Objectif principal

Par équipe de 4 , nous devons concevoir la géométrie d'un avion de ligne capable d'effectuer le trajet **Lille-Copenhague** (avec 150 passagers) en consommant le moins de carburant possible.

1.2 Outils et méthodologie

Le challenge est purement numérique et multidisciplinaire :

— **Aérodynamique & géométrie :** Utilisation de XFLR5 pour concevoir l'avion et estimer les coefficients (le fuselage est ignoré pour l'aéro mais compté dans la masse).

— **Structure :** Utilisation et amélioration d'un outil Python pour vérifier que le fuselage et les ailes résistent aux efforts (RDM).

— **Dynamique du vol :** Utilisation d'un simulateur Python pour tester la trajectoire.

— **Gestion :** Estimation des coûts et de la chaîne de valeur.

1.3 La mission (Simulation)

La performance est évaluée sur une simulation de 200 secondes extrapolée pour le trajet complet. Le départ se fait en vol droit à altitude de croisière h, cap -45° (Nord). Une manœuvre de virage doit être effectuée pour atteindre le cap -60° (vers Copenhague). Les conditions de réussite sont d'arriver au cap objectif ($|\Psi + 60°| < 1°$) et de maintenir l'altitude et l'angle de dérapage stables.

2 Répartition du travail

Le tableau ci-dessous résume la répartition des tâches au sein de l'équipe :

Membre	**Responsabilités**
Baptiste	Interprétation Python de la partie structure (calculs de moments, déplacements, forces).
Jules	Réalisation de la simulation sur XFLR5 et analyse.
Inès	Utilisation du simulateur.
David	Rédaction du rapport final.

TABLE 1 – Répartition des rôles

Nous avons créé un repository GitHub ainsi qu'un Drive pour la collaboration.

3 Déroulé du projet : Phases 1 et 2

La première phase a consisté en la définition du projet et la mise en équations. La deuxième phase s'est concentrée sur la finalisation du rapport intermédiaire et la répartition du travail. Ces éléments ont été détaillés dans le rapport précédent.

Le script Python initial permettait déjà de calculer les propriétés de section, le moment de flexion, le déplacement et la force critique de flambage.

4 Phase 3 : Dimensionnement Python (Structure)

Cette phase est consacrée à l'évaluation de la MTOW (Maximum Take-Off Weight) et à la vérification de la réalisabilité structurelle des ailes et du fuselage via trois programmes Python interconnectés.

4.1 Programme de dimensionnement de voilure (`wing_opti.py`)

4.1.1 Objectif et approche

Le script `wing_opti.py` est un outil d'estimation de masse qui confronte deux approches :

1. **Approche empirique :** Basée sur des statistiques et les équations de Kroo (2001) et Torenbeek (1986).
2. **Approche physique :** Basée sur l'optimisation structurelle d'un caisson de voilure soumis à des contraintes de Résistance des Matériaux (RDM).

Le matériau utilisé est un Aluminium 7075/2024 ($E = 71.7$ GPa, $\sigma_{yield} = 450$ MPa, $\rho = 2800$ kg/m^3) avec un facteur de sécurité $SF = 1.5$.

4.1.2 Optimisation structurelle (Code)

Le cœur du programme modélise l'emplanture comme un caisson rectangulaire raidi. L'algorithme SLSQP minimise la masse sous contrainte de Von Mises et de flambage.

Pour le flambement de la peau (Skin Buckling), nous avons implémenté la formule exacte pour une plaque rectangulaire simplement supportée, en tenant compte du coefficient de Poisson (ν) :

$$\sigma_{cr} = K_c \frac{E\pi^2}{12(1-\nu^2)} \left(\frac{t}{b}\right)^2 \tag{1}$$

Avec $K_c = 4.0$, E le module d'Young, t l'épaisseur de la peau et b l'espacement entre les raidisseurs.

Voici l'extrait du code mettant en œuvre ce calcul :

```python
def check_structure_constraints_scaled(x_scaled, params):
    # [...] Recuperation des variables

    # 1. Inertie
    Area_tot = Area_skin + Area_str_tot
    Ixx = 2 * (Area_tot * (h_box / 2)**2)

    # 2. Contrainte Flexion
    sigma_f = (M_f * (h_box / 2)) / Ixx

    # 3. Marges
    # a) Yield
    margin_yield = (YIELD_STRESS / (SAFETY_FACTOR * sigma_f)) - 1.0
```

```
14
15      # b) Buckling Skin (Formule exacte)
16      spacing = w_box / (n_str + 1)
17      Kc = 4.0
18      # Terme incluant le coefficient de Poisson v (0.33)
19      term_pre = (E_MODULUS * np.pi**2) / (12 * (1 - POISSON_RATIO**2))
20      sigma_cr_skin = Kc * term_pre * (t_skin / spacing)**2
21
22      margin_skin = (sigma_cr_skin / (SAFETY_FACTOR * sigma_f)) - 1.0
23
24      # c) Buckling Stringer (Euler)
25      F_cr_str = (np.pi**2 * E_MODULUS * I_str_own) / (L_rib**2)
26      # [...]
27      return [margin_yield, margin_skin, margin_str]
```

Listing 1 – Extrait de `wing_opti.py` : Vérification des contraintes

4.2 Programme de dimensionnement de fuselage (`fuselage_opti.py`)

4.2.1 Objectif et cas de charge

Ce script estime la masse du fuselage. Contrairement à l'aile, le chargement dimensionnant inclut la **pressurisation cabine** ($\Delta P = 60000$ Pa) combinée à la flexion due à l'empennage.

Le fuselage est modélisé comme une coque cylindrique raidie. L'épaisseur totale travaillante est la somme de l'épaisseur de peau et de l'épaisseur équivalente des raidisseurs ($t_{total} = t_{peau} + t_{eq\_str}$).

4.2.2 Calcul des contraintes (Code)

Le programme vérifie le critère de Von Mises combinant les contraintes circonférentielles (hoop stress) et longitudinales.

```
1  def calcul_contraintes_fuselage(x, params):
2      # [...]
3      # 1. Pression
4      sigma_hoop = (P_int * R) / t_skin
5      t_total = t_skin + t_eq_str
6      sigma_long_p = (P_int * R) / (2 * t_total)
7
8      # 2. Flexion
9      sigma_bend = (M_bend * R) / I_fus
10
11     # 3. Combinaison (Von Mises)
12     sigma_long_tot = sigma_long_p + sigma_bend
13     vm_stress = np.sqrt(sigma_hoop**2 + sigma_long_tot**2 - sigma_hoop *
       sigma_long_tot)
14
15     margin_yield = (YIELD_STRESS / (SAFETY_FACTOR * vm_stress)) - 1.0
16     # [...]
17     return [margin_yield, margin_buckling]
```

Listing 2 – Extrait de `fuselage_opti.py` : Calcul Von Mises

4.3 Programme d'intégration globale (`structure.py`)

Ce script agit comme un intégrateur système. Il valide la faisabilité de la mission Lille-Copenhague en calculant le carburant nécessaire (équation de Bréguet) et le bilan de masse complet.

Une fonctionnalité clé est le calcul du **tenseur d'inertie** de l'avion complet, essentiel pour la simulation de vol. L'aile est traitée par la méthode des bandes (Strip Theory) et transportée au centre de gravité global via le théorème de Huygens.

```
def calcul_inertie_globale(m_comps, geo):
    # [...] Boucle sur les elements
        # Transport Huygens standard
        Ixx_tot += di_xx + dm * (dy**2 + dz**2)
        Iyy_tot += di_yy + dm * (dx**2 + dz**2)
        Izz_tot += di_zz + dm * (dx**2 + dy**2)
        Ixz_tot += dm * (dx * dz)

    # Transport au CG Avion final
    Ixx_cg = Ixx_tot - M_tot * (Z_CG**2)
    Iyy_cg = Iyy_tot - M_tot * (X_CG**2 + Z_CG**2)
    Izz_cg = Izz_tot - M_tot * (X_CG**2)

    return [Ixx_cg, Iyy_cg, Izz_cg, Ixy_cg, Iyz_cg, Ixz_cg], (X_CG, Z_CG
    )
```

Listing 3 – Extrait de `structure.py` : Calcul Inertie Globale

4.4 Essais

Ces tableaux récapitulent tous les essais réalisés sur les 4 types d'avions étudiés (tous les résultats des analyses sont enregistrés sous format texte dans le dossier src/analysis).

4.4.1 Etude de l'A320-200

Optimisation de l'aile :

Masse struct. brute	Ep. revêtement	Nbr. lisses	Section lisse (HxL)	Masse totale ailes
8310.2 kg	4.73 mm	29	89.9 mm x 13.4 mm	11218.8 kg

Optimisation du fuselage :

Masse struct. brute	Ep. revêtement	Masse totale fuselage
6664.7 kg	4.13 mm	9330.6 kg

4.4.2 Etude de l'A330-300

Optimisation de l'aile :

Masse struct. brute	Ep. revêtement	Nbr. lisses	Section lisse (HxL)	Masse totale ailes
44991.1 kg	5.28 mm	30	101.1 mm x 42.2 mm	60737.9 kg

Optimisation du fuselage :

Masse struct. brute	Ep. revêtement	Masse totale fuselage
29584.1 kg	7.76 mm	41417.7 kg

4.4.3 Etude du B747-400

Optimisation de l'aile :

Masse struct. brute	Ep. revêtement	Nbr. lisses	Section lisse (HxL)	Masse totale ailes
69906.7 kg	8.08 mm	31	89.3 mm x 62.8 mm	94374.0 kg

Optimisation du fuselage :

Masse struct. brute	Ep. revêtement	Masse totale fuselage
50734.8 kg	10.44 mm	71028.8 kg

4.4.4 Etude du B777-200

Optimisation de l'aile :

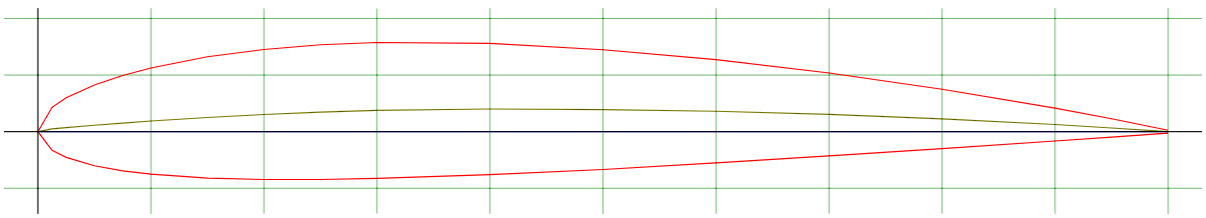
Masse struct. brute	Ep. revêtement	Nbr. lisses	Section lisse (HxL)	Masse totale ailes
43937.8 kg	9.54 mm	30	95.0 mm x 27.4 mm	59316.0 kg

Optimisation du fuselage :

Masse struct. brute	Ep. revêtement	Masse totale fuselage
41816.6 kg	8.62 mm	58543.2 kg

5 Phase 4 : Modélisation XFLR5 et extraction de données

Nous avons étudié quatre avions, tous des avions de ligne. Tous sont issus du fichier "Aircraft.Data.xlsx" Les avions retenus sont l'A320-200, l'A330-300, le B747-400 et le B777-200. Tous ont le profil d'ailes NACA 2412.



5.1 Étude sur l'A320-200

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	230 m/s	33.9 m	21.71°	60 000 kg

Résultats analyse Python :

Inertie (kg.m^2)	I_{xx}	I_{yy}	I_{zz}
Valeur	1.82×10^6	3.85×10^6	5.53×10^6

La surface d'aile obtenue via Python est 136.28 m^2

Analyse XFLR5 : La surface d'aile obtenue est de 128.82 m^2.
— $\alpha_{op} = -3.48°$
— $dm_{op} = -0.0927/°$

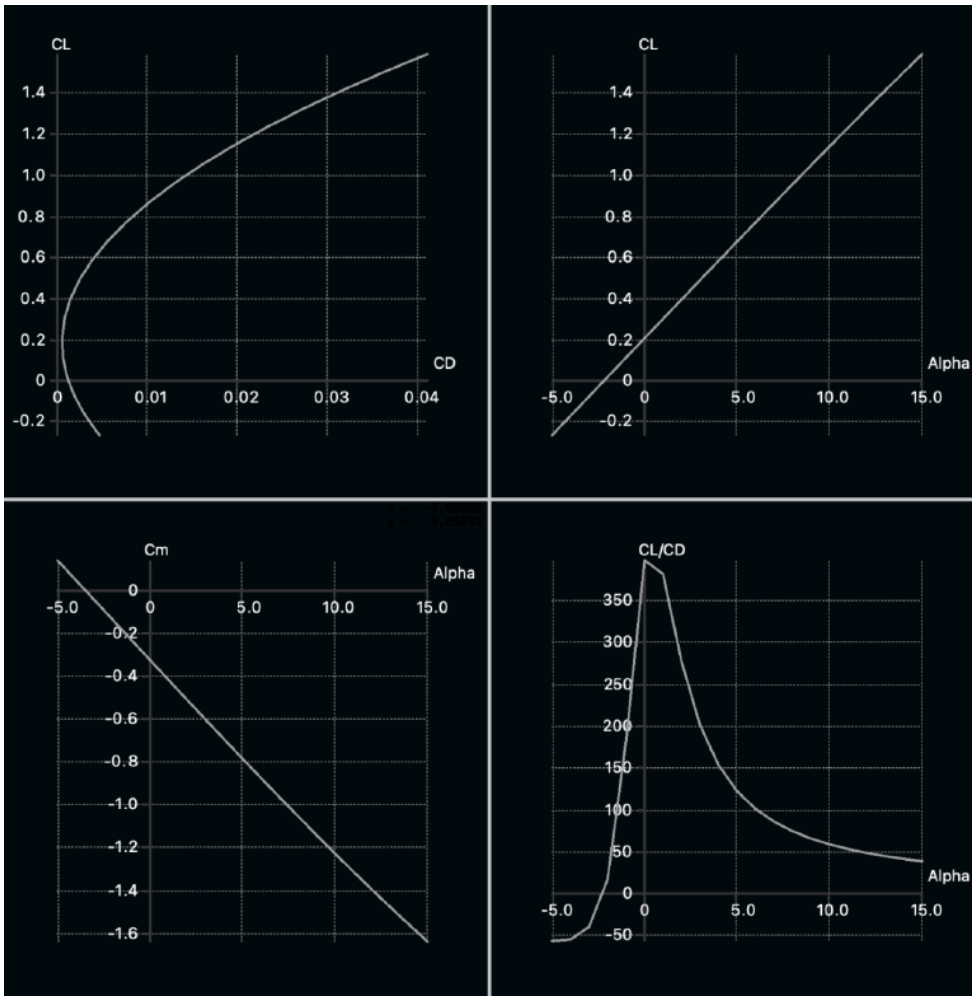


FIGURE 1 – Courbes polaires A320 (Source : XFLR5)

Control et stability derivatives déduites (A320-200) :

Coef	_0 (Trim)	_alph (α)	_beta (β)	_p (Roll)	_q (Pitch)	_r (Yaw)	_dl (Ail)	_dm (Elev)	_dn (Rud)
Cxs	0.0240	0.150	0	0	0	0	0	0	0
Cys	0	0	-0.910	0	0	0	0	0	0.16
CL	0.200	5.650	0	0	6.800	0	0	0.38	0
Cl	0	0	-0.120	-0.450	0	0.110	0.08	0	0.015
Cm	0	-0.950	0	0	-20.500	0	0	-1.35	0
Cn	0	0	0.145	-0.060	0	-0.180	-0.005	0	-0.11

TABLE 2 – Coefficients aérodynamiques représentatifs du A320-200

5.2 Étude sur l'A330-300

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	230 m/s	60.3 m	30°	212 000 kg

Résultats analyse Python :

Inertie (kg.m^2)	I_{xx}	I_{yy}	I_{zz}
Valeur	2.16×10^7	3.17×10^7	5.24×10^7

La surface d'aile obtenue via Python est 339.46 m^2

Analyse XFLR5 : La surface d'aile obtenue est de 362 m^2.
— $\alpha_{op} = -1.19°$
— $dm_{op} = -0.06/°$

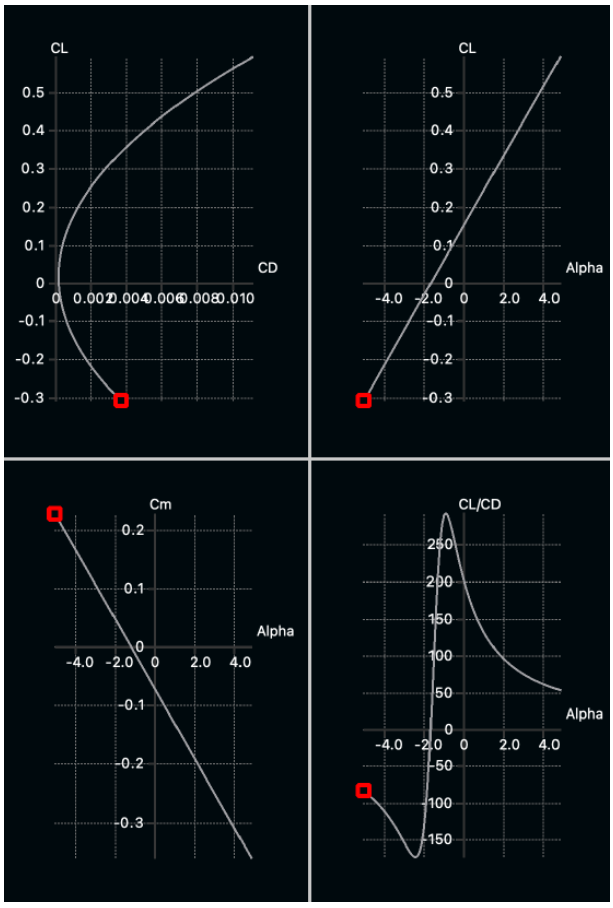


FIGURE 2 – Courbes polaires A330 (Source : XFLR5)

Control et stability derivatives déduites :

Coef	_0 (Trim)	_alph (α)	_beta (β)	_p (Roll)	_q (Pitch)	_r (Yaw)	_dl (Ail)	_dm (Elev)	_dn (Rud)
Cxs	0.0027	0.199	0	0	0	0	0.01	0.02	0.02
Cys	0	0	-0.365	0.056	0	0.241	0	0	0.20
CL	0.296	5.250	0	0	7.367	0	0	0.35	0
Cl	0	0	-0.056	-0.484	0	0.083	0.15	0	0.01
Cm	0	-0.513	0	0	-19.243	0	0	-1.20	0
Cn	0	0	0.111	-0.043	0	-0.073	-0.01	0	-0.12

TABLE 3 – Coefficients aérodynamiques du A330

5.3 Étude sur le B747-400

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	230 m/s	62.3 m	38°	300 000 kg

Résultats analyse Python :

Inertie (kg.m^2)	I_{xx}	I_{yy}	I_{zz}
Valeur	4.00×10^7	6.85×10^7	1.06×10^8

La surface d'aile obtenue via Python est 516.31 m²

Analyse XFLR5 : La surface d'aile obtenue est de 525 m².
— $\alpha_{op} = -0.79°$
— $dm_{op} = -0.0516/°$

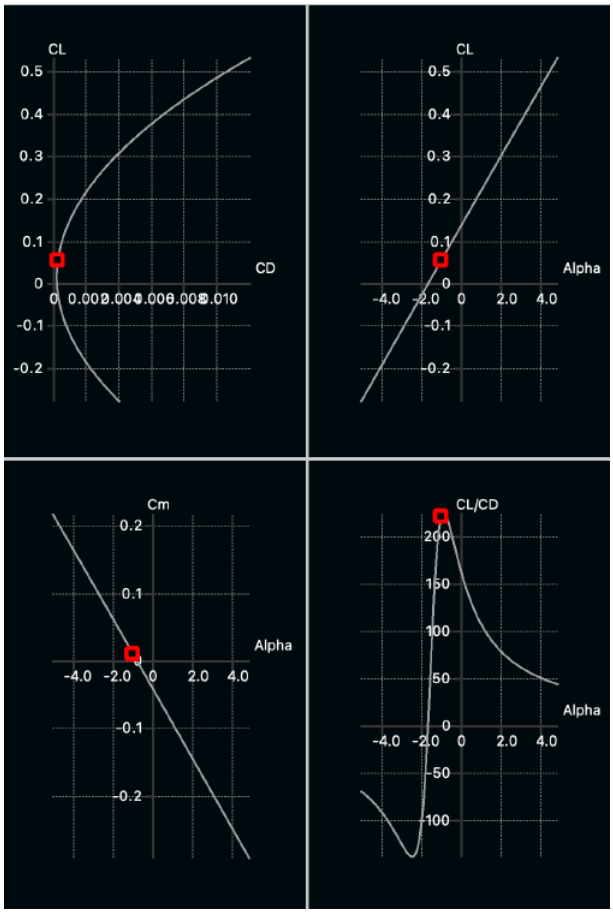


FIGURE 3 – Courbes polaires B747 (Source : XFLR5)

Control et stability derivatives déduites (B747-400) :

Coef	_0 (Trim)	_alph (α)	_beta (β)	_p (Roll)	_q (Pitch)	_r (Yaw)	_dl (Ail)	_dm (Elev)	_dn (Rud)
Cxs	0.0400	0.120	0	0	0	0	0	0	0
Cys	0	0	-0.870	0	0	0	0	0	0.115
CL	0.650	5.500	0	0	5.200	0	0	0.34	0
Cl	0	0	-0.160	-0.340	0	0.130	0.014	0	0.008
Cm	0	-0.350	0	0	-20.800	0	0	-1.30	0
Cn	0	0	0.130	-0.025	0	-0.190	-0.002	0	-0.105

TABLE 4 – Coefficients aérodynamiques du Boeing 747-400

5.4 Étude sur le B777-200

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	230 m/s	62.9 m	31.6°	200 000 kg

Résultats analyse Python :

Inertie (kg.m²)	I_{xx}	I_{yy}	I_{zz}
Valeur	2.41×10^7	5.29×10^7	7.56×10^7

La surface d'aile obtenue via Python est 495.9 m²

Analyse XFLR5 : La surface d'aile obtenue est de 428 m².
— $\alpha_{op} = -0.92°$
— $dm_{op} = -0.0611/°$

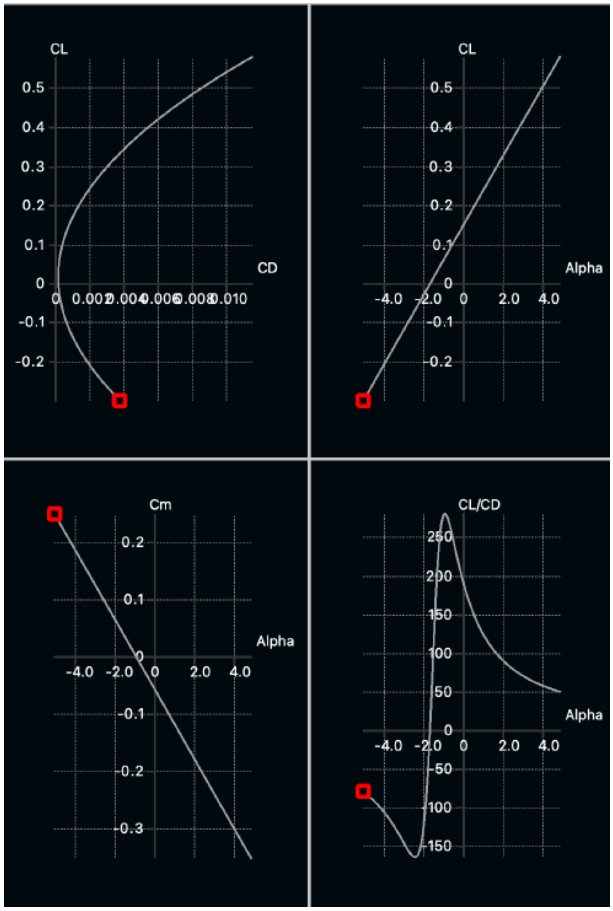


FIGURE 4 – Courbes polaires B777 (Source : XFLR5)

Control et stability derivatives déduites :

Coef	_0 (Trim)	_alph (α)	_beta (β)	_p (Roll)	_q (Pitch)	_r (Yaw)	_dl (Ail)	_dm (Elev)	_dn (Rud)
Cxs	0.00063	0.082	0	0	0.013	0	0.01	0.02	0.02
Cys	0	0	-0.353	-0.107	0	0.347	0	0	0.20
CL	0.129	5.073	0	0	8.936	0	0	0.35	0
Cl	0	0	-0.104	-0.464	0	0.066	0.15	0	0.01
Cm	0	-1.196	0	0	-27.037	0	0	-1.20	0
Cn	0	0	0.153	-0.015	0	-0.156	-0.01	0	-0.12

TABLE 5 – Coefficients aérodynamiques du B777

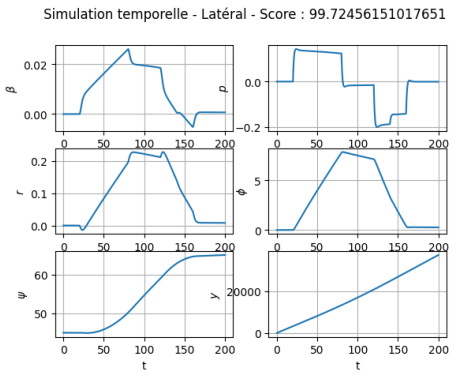
6 Phase 5 : Simulation

Tous les résultats ci-dessous sont trouvables dans le dossier : `Vad_grp6/plane_score/plane_name_results`

6.1 Étude sur l'A320-200

t_i	dl	dm	dn	T
0	0	-1.9	0	36600.0
20	0.07	-1.9	0	36600.0
40	0.07	-1.9	0	36600.0
60	0.07	-1.9	0	36600.0
80	0	-1.9	0	36600.0
100	0	-1.9	0	36600.0
120	-0.09	-1.9	0	36600.0
140	-0.07	-1.9	0	36600.0
160	0	-1.9	0	36600.0
200	0	-1.9	0	36600.0

(a) Vue simulateur



(b) Vue graphique

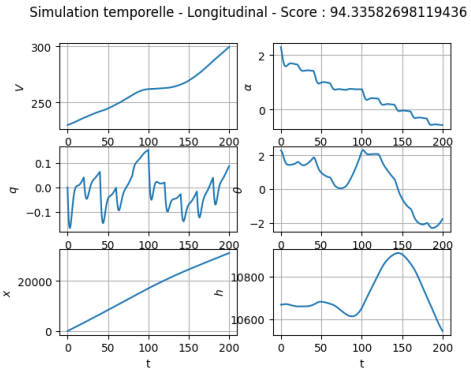
FIGURE 5 – Comparaison des résultats du simulateur A320

Score obtenu : 99.7246

6.2 Étude sur l'A330-300

t_i	dl	dm	dn	T
20	0	-1.2	0	120000.0
40	0.05	-1	0	120000.0
60	0.05	-0.9	0	120000.0
80	0	-0.92	0	120000.0
100	0	-0.75	0	120000.0
120	0	-0.65	0	120000.0
140	-0.06	-0.55	0	120000.0
160	-0.02	-0.45	0	120000.0
180	0	-0.35	0	120000.0
200	0	-0.35	0	120000.0

(a) Vue simulateur



(b) Vue graphique

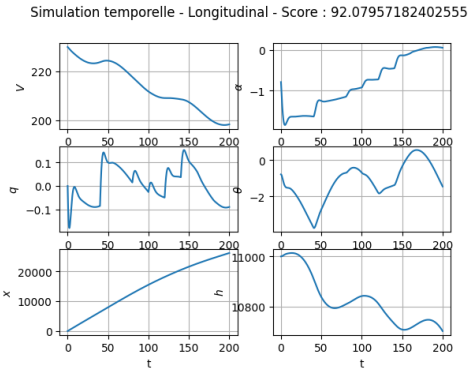
FIGURE 6 – Comparaison des résultats du simulateur A330

Score obtenu : 94.3358

6.3 Étude sur le B747-400

t_i	dl	dm	dn	T
20	0	0.45	0	210000
40	0.9	0.3	0	210000
60	0.9	0.3	0	210000
80	0.9	0.25	0	210000
100	0	0.2	0	210000
120	0	0.1	0	210000
140	-0.5	0	0	210000
160	-1	0	0	210000
180	-0.8	0	0	210000
200	0	0	0	210000

(a) Vue simulateur



(b) Vue graphique

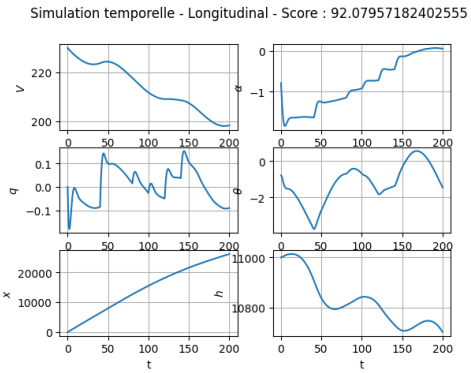
FIGURE 7 – Comparaison des résultats du simulateur B747

Score obtenu : 92.0796

6.4 Étude sur le B777-200

t_i	dl	dm	dn	T
20	0	-2.225	0	155000.0
40	0	-2.225	0	155000.0
60	0.11	-2.23	0	155000.0
80	0.11	-2.21	0	155000.0
100	0	-2.215	0	155000.0
120	0	-2.215	0	155000.0
140	-0.11	-2.215	0	155000.0
160	-0.11	-2.1	0	155000.0
180	0	-2.19	0	155000.0
200	0	-2.19	0	155000.0

(a) Vue simulateur



(b) Vue graphique

FIGURE 8 – Comparaison des résultats du simulateur B777

Score obtenu : 95.3580

7 Phase 6 : Etude Matlab

Tous les résultats ci-dessous sont trouvables dans le dossier `Vad_grp6/matlab`

7.1 Premier modèle : monovariable

7.1.1 Définition du système

Ce modèle correspond au mouvement à courte période. Les variables d'état sont l'incidence et la vitesse de tangage :

$$\vec{X_0} = (\alpha_0, q_0) = (0.01745, 0) \quad \text{avec} \quad \vec{U_0} = (0)$$

Les matrices d'état du système linéarisé sont :

$$A = \begin{pmatrix} -0.488 & 0.993 \\ -13.3 & -1.50 \end{pmatrix} \quad B = \begin{pmatrix} -8.76 \times 10^{-4} \\ -0.273 \end{pmatrix}$$

Les valeurs propres (modes naturels) du système sont :

$$\lambda_{1,2} = -0.99475956 \pm 3.60019425j$$

7.1.2 Commande par retour d'état

La loi de commande est définie par $u = -Kx + Gv$. Pour le placement de pôles, nous avons choisi une dynamique deux fois plus rapide que la partie réelle naturelle :

$$\text{Poles}_{BF} = [-2 + 3j, \quad -2 - 3j]$$

Les gains calculés sont :
— Gain de retour : $K = \begin{pmatrix} 6.7456 & -7.3916 \end{pmatrix}$
— Gain de pré-compensation : $G = -47.7234$
Le système en boucle fermée est alors défini par :

$$sys_{cl} = ss(A - B \cdot K, B \cdot G, C, D)$$

7.1.3 Observateur de Luenberger

L'équation de l'observateur est donnée par :

$$\frac{d(\hat{x})}{dt} = A\hat{x} + Bu + L(y - C\hat{x})$$

Nous choisissons des pôles d'observateur 5 fois plus rapides que ceux du système en boucle fermée (partie réelle) :

$$\text{Poles}_{Obs} = [-10, \quad -11]$$

Le gain de l'observateur L obtenu est :

$$L = \begin{pmatrix} 19.0120 \\ 68.0192 \end{pmatrix} \begin{matrix} \text{(Correction sur } \alpha\text{)} \\ \text{(Correction sur } q\text{)} \end{matrix}$$

7.1.4 Commande intégrale

L'intégrateur est ajouté pour annuler l'erreur statique face aux perturbations constantes. On augmente le système avec un nouvel état $\dot{\xi} = r - y$. Le pôle de l'intégrateur est placé en -5.

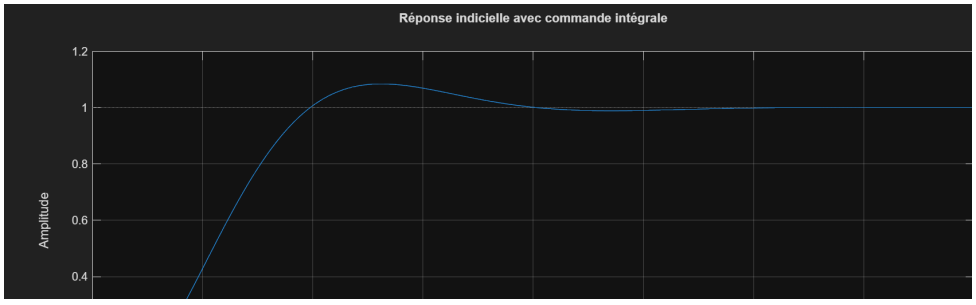


FIGURE 9 – Réponse indicielle avec commande intégrale (modèle monovariable)

7.2 Second modèle : multivariable

7.2.1 Définition du système

$$\vec{X_0} = (V_0, \alpha_0, q_0, \theta_0) = (283, 0.01745, 0, 0.01745)$$
$$\vec{U_0} = (0.26000)$$

Les matrices du système sont :

$$A = \begin{pmatrix} -0.00255 & 6.26 & -0.0486 & -9.81 \\ -0.000247 & -0.488 & 0.993 & 0 \\ 0 & -13.3 & -1.50 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} -0.00637 & 1.37 \times 10^{-8} \\ -0.00087 & -8.46 \times 10^{-10} \\ -0.273 & 0 \\ 0 & 0 \end{pmatrix}$$

Les modes propres sont :
— Mode 1 : $\lambda_{1,2} = -0.9948 \pm 3.6001j$
— Mode 2 : $\lambda_{3,4} = -0.0012 \pm 0.0481j$

7.2.2 Commande par retour d'état

Objectif : couplage vitesse-pente accéléré (-0.05) et courte période amortie (-2).

$$\text{Poles}_{BF} = [-0.05 \pm 0.1j, \quad -2.0 \pm 2.0j]$$

Gain de retour K (2×4) obtenu (facteur 10^6) :

$$K \approx 10^6 \times \begin{pmatrix} 0.0000 & 0.0000 & -0.0000 & -0.0000 \\ 0.7274 & 3.0844 & 0.3772 & -7.3081 \end{pmatrix}$$

7.2.3 Observateur d'état

Les pôles de l'observateur sont fixés à : $[-0.5, -0.6, -10, -12]$. Le gain L (4×2) calculé est :

$$L = \begin{pmatrix} 12.1862 & -4.0187 \\ -1.9143 & 8.6435 \\ -22.4039 & -21.1844 \\ -2.0556 & 8.9233 \end{pmatrix}$$

7.2.4 Commande intégrale

On ajoute deux intégrateurs pour le suivi de consigne (vitesse et assiette).
— Gain proportionnel K_p (sur les états) :

$$K_p \approx 10^6 \times \begin{pmatrix} -0.0000 & 0.0001 & -0.0000 & -0.0000 \\ 1.1058 & 4.5288 & -0.7052 & -5.9395 \end{pmatrix}$$

— Gain intégral K_i (sur les erreurs) :

$$K_i \approx 10^5 \times \begin{pmatrix} 0.0000 & 0.0000 \\ -2.4957 & 0.0849 \end{pmatrix}$$

7.2.5 Simulation

Simulation d'une consigne de +10 m/s sur la vitesse V et 0 sur l'assiette θ.

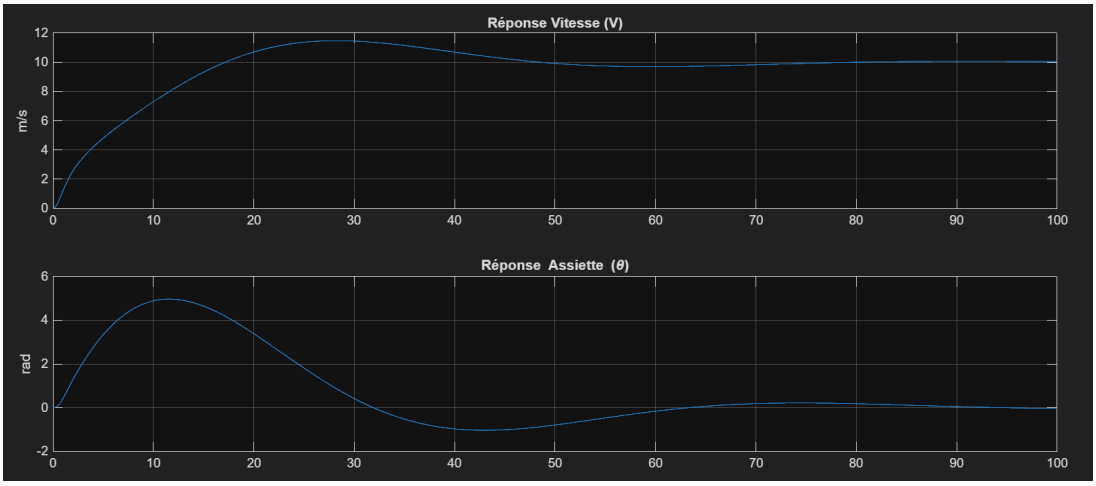


FIGURE 10 – Simulation : Réponse vitesse (haut) et assiette (bas)

8 Conclusion et ouverture

8.1 Résultats et interprétation

Concernant les ailes et leur optimisation, les résultats de xflr5 et du script python sont cohérents avec les données issues de la littérature. Connaissant les paramètres d'un profil d'aile, le revêtement et les lisses ont pu être optimisés.

Le fuselage a également pu être optimisé pour satisfaire les conditions de pression, du flambage et du poids des composants.

Finalement, la MTOW calculée et la masse de l'avion ont pu être déduits de ces informations.

Les scores obtenus pour les différents avions dépassent tous 90 ce qui peut être considéré comme une valeur acceptable.

8.2 Estimation des coûts, gestion et chaîne de valeur

Bien que ce challenge soit purement numérique, une analyse de la viabilité économique est essentielle pour valider le projet industriel.

- **Gestion des ressources et coûts matières :** Le choix de l'Aluminium 7075/2024 ($\sigma_{yield} = 450$ MPa) permet de concilier une haute résistance mécanique et un coût de fabrication maîtrisé par rapport aux composites. La minimisation de la masse structurale brute (ex : 8310.2 kg pour l'A320) via l'optimisation Python réduit directement l'investissement en matières premières.
- **Rentabilité opérationnelle :** Avec un $TSFC$ (consommation spécifique) imposé de $15\frac{g}{kNs}$, la réduction de la masse totale via les scripts `wing_opti.py` et

`fuselage_opti.py` est le levier principal pour minimiser la perte de carburant sur le trajet Lille-Copenhague.

— **Chaîne de valeur :** L'utilisation d'outils basse-fidélité comme XFLR5 et de l'intégration système sous Python permet de réduire drastiquement les coûts de R&D lors de la phase de conception préliminaire.

— **Organisation :** La répartition des rôles (Structure, Simulation, XFLR5, Rédaction) a permis une parallélisation des tâches, optimisant ainsi le "temps de mise sur le marché" (time-to-market) du concept.

8.3 Esprit critique et limites de l'étude

Il est crucial d'analyser la pertinence des hypothèses sélectionnées pour évaluer la fiabilité de nos résultats.

— **Hypothèses aérodynamiques :** Dans XFLR5, le fuselage a été ignoré pour le calcul des coefficients aérodynamiques, bien que sa masse soit comptabilisée. Cela mène à une sous-estimation de la traînée totale et donc à une estimation trop optimiste de la consommation de carburant.

— **Modélisation de la masse :** Conformément aux hypothèses de l'énoncé, la masse de l'aéronef a été supposée constante durant la simulation. En réalité, la consommation de carburant sur un trajet réel allégerait l'avion, modifiant son centre de gravité et sa stabilité longitudinale au fil du vol.

— **Limites de la structure :** Le matériau a été considéré comme homogène. L'intégration de matériaux composites pour la peau, combinée à des raidisseurs en alliage, pourrait offrir un gain de masse supplémentaire par rapport à notre solution actuelle.

— **Analyse des scores :** Bien que les scores de simulation dépassent 90 (ex : 99.72 pour l'A320), la précision de la tenue d'altitude lors du virage vers le cap $-60°$ dépend fortement du réglage des contrôleurs de vitesse et de dérapage, qui pourraient être encore affinés.

8.4 Conclusion

Ce projet a permis d'intégrer les disciplines de structure (RDM), d'aérodynamique et de mécanique du vol. L'utilisation conjointe de scripts Python pour le dimensionnement massique et de XFLR5 pour l'analyse aérodynamique a abouti à une conception approchée mais cohérente du problème.