VERS UNE AVIATION DURABLE

Rapport Final

Auteurs :

Lefebvre Inès
Neveux Baptiste
Larrouturou David
Leene Jules

23 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	Rédaction du rapport final.
Inès	Utilisation du simulateur.
David	Réalisation de la simulation sur XFLR5 et analyse.

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.

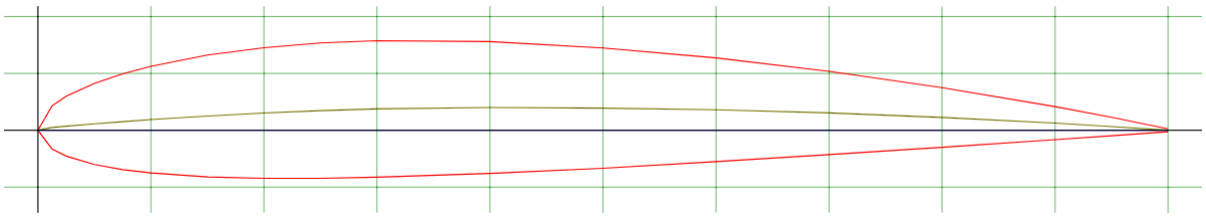
Une fonctionnalité clé est le calcul du **tenseur d'inertie** de l'avion complet, essentiel pour la simulation de vol. L'aile est traitée par la méthode des bandes (Strip Theory) et transportée au centre de gravité global via le théorème de Huygens.

```
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

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	150 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.
— $\alpha_{op} = -3.48°$
— $dm_{op} = -0.0927/°$

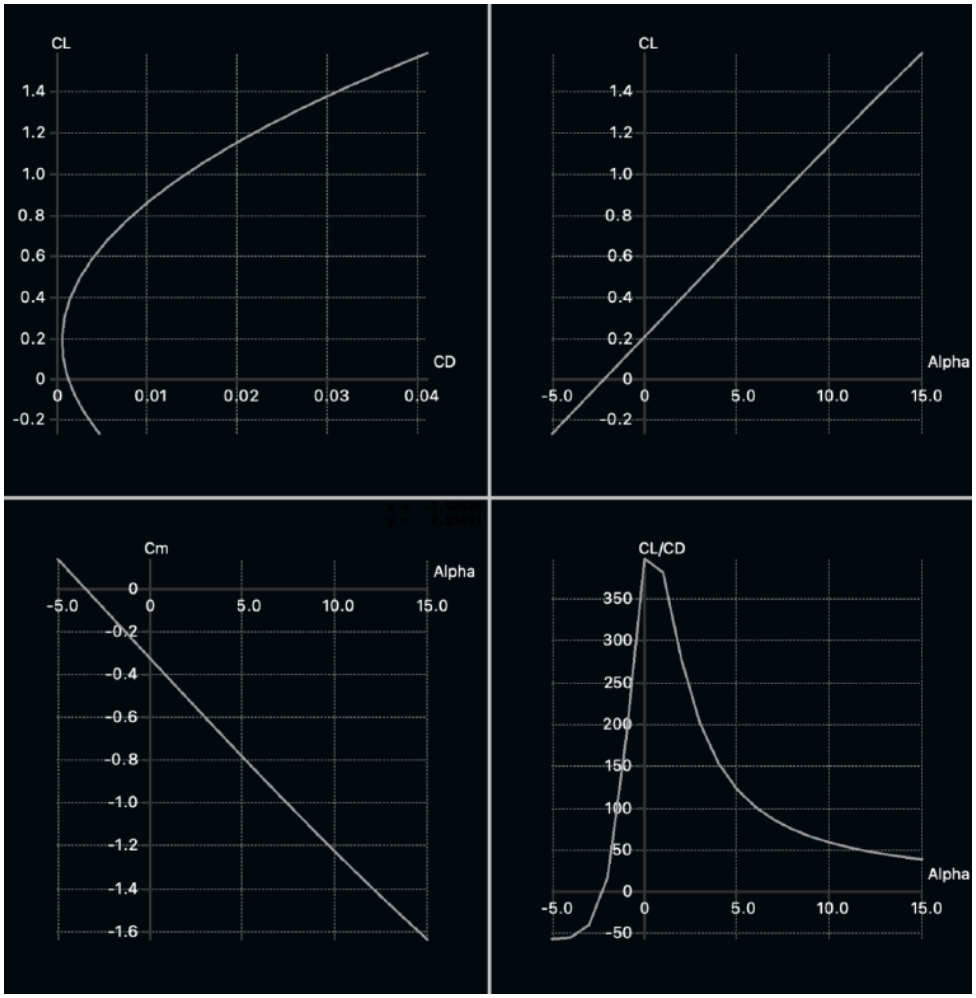


FIGURE 1 – Courbes polaires A320 (Source : XFLR5)

5.2 Étude sur l'A330-300

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	150 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.
— $\alpha_{op} = -1.19°$
— $dm_{op} = -0.06/°$

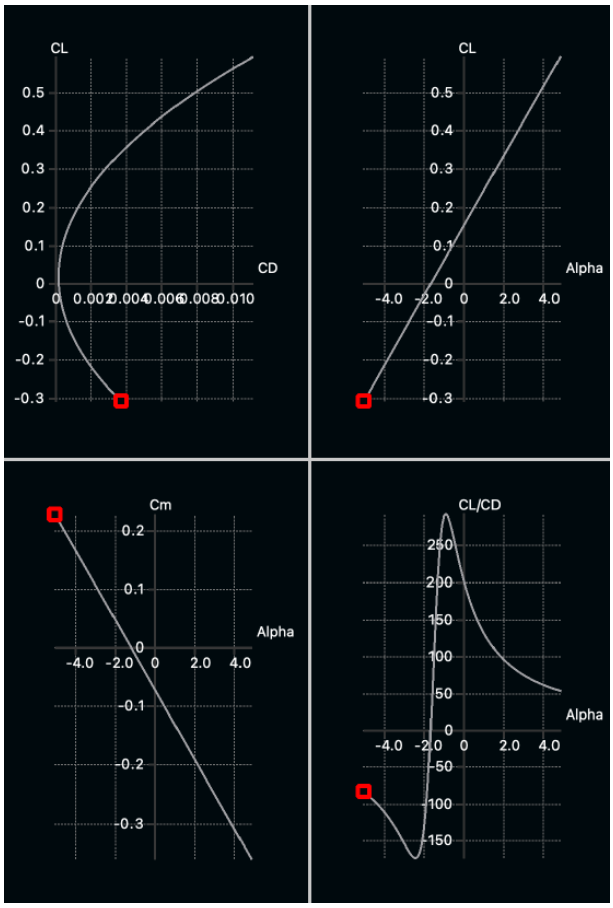


FIGURE 2 – Courbes polaires A330 (Source : XFLR5)

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 2 – Coefficients aérodynamiques du A330

5.3 Étude sur le B747-400

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	150 m/s	62.3 m	38°	300 000 kg

Résultats analyse Python :

Inertie (kg.m²)	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.
— $\alpha_{op} = -0.79°$
— $dm_{op} = -0.0516/°$

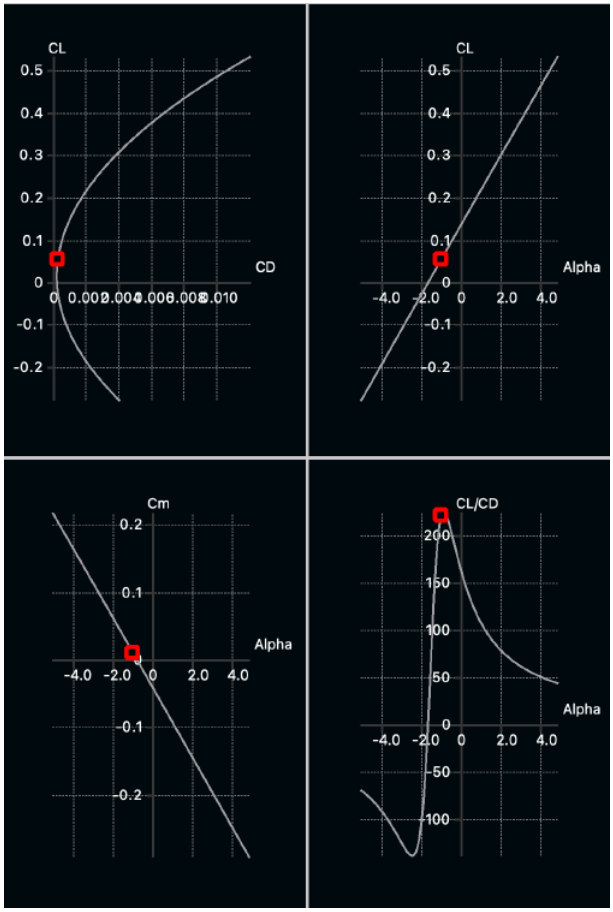


FIGURE 3 – Courbes polaires B747 (Source : XFLR5)

5.4 Étude sur le B777-200

Caractéristiques :

Profil	V_∞	Envergure	Sweep	Masse
NACA 2412	150 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.
— $\alpha_{op} = -0.92°$
— $dm_{op} = -0.0611/°$

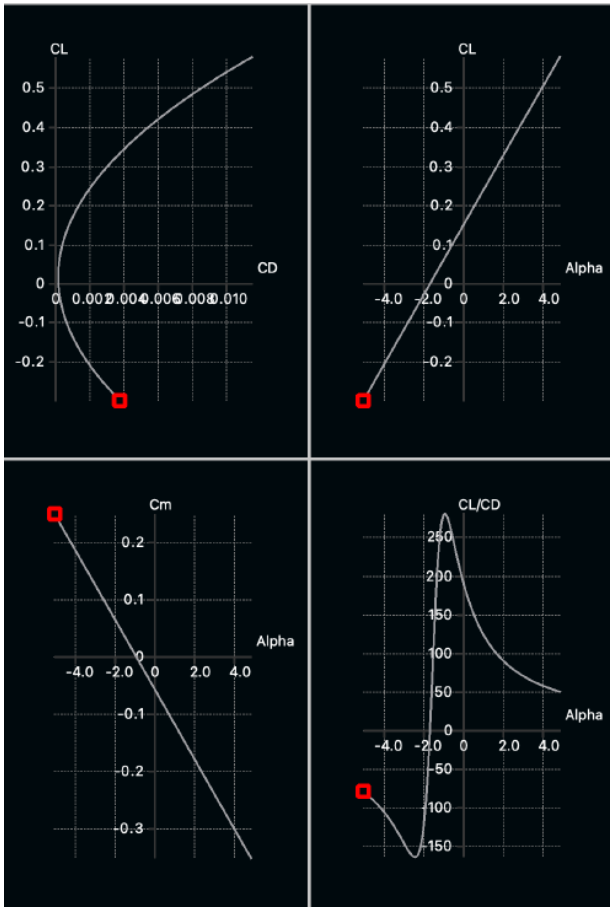


FIGURE 4 – Courbes polaires B777 (Source : XFLR5)

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 3 – Coefficients aérodynamiques du B777

6 Phase 5 : Simulation

Partie en cours de réalisation, nous n'arrivons pas à faire marcher le simulateur

7 Conclusion et ouverture

7.1 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.

Pour l'instant, nous avons des difficultés à tester le simulateur. Après plusieurs essais, le score est toujours cantonné à 0.

7.2 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.