Vers une Aviation Durable

Rapport Intermédiaire

Lefebvre Inès, Neveux Baptiste, Larrouturou David, Leene Jules

5 décembre 2025

Table des matières

1 Définition du projet

1.1 Objectif Principal

Le projet consiste à simuler une entreprise (équipe de 4) devant concevoir la géométrie d'un avion de ligne capable d'effectuer le trajet **Lille-Copenhague** avec 150 passagers (soit environ 13.5 tonnes de masse utile, sur la base de 90 kg par personne avec bagages). L'objectif central est de consommer le moins de carburant possible.

1.2 La Mission (Simulation)

La performance de l'avion est évaluée sur une simulation numérique de 200 secondes, extrapolée ensuite pour le trajet complet. Le profil de la mission est le suivant :

- **Départ** : Vol droit à altitude de croisière $h = 25\,000$ m et vitesse de croisière $V_c = 870$ km/h, avec un cap initial de -45°(Nord).
- **Manœuvre** : Virage pour atteindre le cap objectif de -60°(direction de Copenhague).
- **Conditions de réussite** :
  - Arriver au cap objectif avec une précision de $|\Psi + 60°| < 1°$.
  - Maintenir l'altitude et l'angle de dérapage stables (pente $< 0.5°$).

1.3 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'aérodynamique mais compté dans la masse).
- **Structure** : Utilisation et amélioration d'un outil Python fourni pour vérifier la résistance du fuselage et des ailes (MM_RDM).
- **Dynamique du vol** : Utilisation d'un simulateur Python pour tester la trajectoire.
- **Gestion** : Estimation des coûts et de la chaîne de valeur.

1.4 Hypothèses simplificatrices

La masse est considérée constante (malgré la consommation), le centre de gravité est fixe, l'atmosphère standard est constante, et le TSFC (Thrust Specific Fuel Consumption) est fixé à 15 g/k.N.s. Pour que la portance compense le poids, la relation suivante doit être vérifiée :

$$L = \frac{1}{2}\rho S V^2 C_L = mg \tag{1}$$

2 Organisation et Planification

2.1 Répartition du travail

Les responsabilités ont été distribuées comme suit au sein de l'équipe :

2.2 Déroulé du projet

- **1ère phase (jusqu'au 28/11)** : Définition du projet, interprétation des consignes, découverte de XFLR5, définition des premiers jalons et mise en équation.
- **2ème phase (jusqu'au 5/12)** : Finalisation du rapport intermédiaire, développement du programme Python et calculs structurels.

Membre	Tâches principales
Baptiste	Interprétation Python de la partie structure (calculs de moments, déplacements, forces...)
Jules	Rédaction du rapport final
Inès	Interprétation des résultats aérodynamiques obtenus
David	Réalisation de la simulation sur XFLR5 et obtention des résultats

TABLE 1 – Matrice de répartition des tâches

3 Programme Python : Analyse Structurelle

Cette section détaille les scripts développés pour répondre aux exercices 1, 2, 3, 4 et 5 du cours de structure. Le code intègre désormais la lecture directe des données avions depuis un fichier Excel (‘Aircraft_Data.xlsx‘).

3.1 Calcul des propriétés d'une section

Le script calcule l'aire (S), les moments quadratiques (I_x, I_y) pour une section en I.

```
def calcul_section(h, b, e):
    """
    Calcule S (Aire), Ix (Inertie flexion), et Iy pour une section en I.
    """
    # Aire : 2*semelles + ame
    S = 2*b*e + h*e
    # Inertie Ix (autour axe neutre horizontal)
    I_x = e*( h**3/12 + b*e**2/6 + b*(h + e)**2/2)
    # Inertie Iy
    I_y = e/6*( h*e**2/2 + b**3)
    return S, I_x, I_y
```

Listing 1 – Fonction calcul_section

Exemple du code et valeurs :
Le test effectué dans le programme utilise les dimensions suivantes : $h = 0.07$ m, $b = 0.03$ m, $e = 0.004$ m.

```
S, Ix, Iy = calcul_section(0.07, 0.03, 0.004)
```

— Aire (S) : 5.200000×10^{-4} m^2
— Moment Quadratique (I_x) : 4.432133×10^{-7} m^4
— Moment Quadratique (I_y) : 1.837333×10^{-8} m^4

3.2 Calcul des moments de flexion et déplacements

Ce module détermine $M(x)$ et $y(x)$ par superposition des cas de charge répartie (q) et de force ponctuelle (F).

```
def calcul_poutre_console_superposition(x_array, L, a, F, q, E, I):
    # --- Cas 1 : Charge répartie q ---
    M_q = -(q / 2) * (L - x_array)**2
    # Formule de la flèche pour charge répartie
    y_q = (q / (24 * E * I)) * x_array**2 * (x_array**2 - 4*L*x_array + 6*L**2)

    # --- Cas 2 : Force ponctuelle F ---
    M_F = np.zeros_like(x_array)
    y_F = np.zeros_like(x_array)

    for i, x in enumerate(x_array):
```

```
        if x <= a:
            M_F[i] = -F * (a - x)
            y_F[i] = (F / (6 * E * I)) * x**2 * (3*a - x)
        else:
            M_F[i] = 0
            y_F[i] = (F / (6 * E * I)) * a**2 * (3*x - a)

    return M_q + M_F, y_q + y_F
```

Listing 2 – Calcul par superposition (C01 + C02)

Exemple du code et valeurs :
Le programme simule une poutre de 20 m avec une force de 200 N placée à 4 m et une charge répartie de -15 N/m.

```
# L=20, a=4, F=200, q=-15
M_res, y_res = calcul_poutre_console_superposition(x, 20, 4, 200, -15, E_ALU, Ix
    )
```

— Moment Fléchissant Max : 2200.00 N.m (à l'encastrement)
— Flèche Maximale (bout de poutre) : 8707 mm

3.3 Force critique de flambage

L'évaluation de la force critique compare le flambage global (Euler) et le voilement local/-plastique (Johnson/Crippling).

```
def calcul_force_critique(L, E, I, S, nu, b, h, e):
    # 1. Flambage Global (Euler)
    K = 0.5
    L_eq = K * L
    if L_eq <= 0: L_eq = 0.001
    F_euler = (np.pi**2 * E * I) / (L_eq**2)

    # 2. Voilement Local (Crippling)
    ratio_bh = b / h
    k_crippling = float(get_k_factor(ratio_bh))
    sigma_cc = ((np.pi * e) / h)**2 * (E / (12 * (1 - nu**2))) * k_crippling

    # 3. Flambage Réel (Johnson)
    rho = np.sqrt(I / S)
    elancement = L_eq / rho

    # Sécurité mathématique
    if sigma_cc <= 0: elancement_critique = 1e9
    else: elancement_critique = np.pi * np.sqrt(2 * E / sigma_cc)

    if elancement >= elancement_critique:
        return F_euler, "Euler (Global)", elancement
    else:
        sigma_johnson = sigma_cc - (sigma_cc**2 * elancement**2) / (4 * np.pi**2
    * E)
        return sigma_johnson * S, "Johnson (Local/Plastique)", elancement
```

Listing 3 – Calcul de la force critique

Exemple du code et valeurs :
Le script teste deux longueurs caractéristiques pour observer le changement de mode de ruine.

```
for L_test in [0.5, 2.0]:
    F_crit, mode, lam = calcul_force_critique(L_test, ...)
```

— Pour $L = 0.5$ m : $F_{crit} \approx 351\,323$ N (Mode Johnson/Plastique)
— Pour $L = 2.0$ m : $F_{crit} \approx 253\,301$ N (Mode Johnson/Plastique)

3.4 Optimisation de la section

Cette fonction itère sur les dimensions (e, b, h) pour trouver la section la plus légère capable de supporter une charge donnée. Le critère fondamental corrigé est que la force critique doit être **supérieure ou égale** à la charge cible.

```python
def optimisation_section(L, F_cible, E, nu):
    solutions = []
    # Boucles sur e (épaisseur), b (largeur), h (hauteur)...
    # ... calcul de S, Ix, Iy et f_max ...

    # CONDITION CRITIQUE : La poutre doit résister (F_max >= Charge)
    if f_max >= F_cible:
        solutions.append({'S': S, 'e': e_eval, 'b': b_eval, 'h': h_eval, 'F_max'
    : f_max})

    if not solutions: return "Aucune section trouvée."

    # On trie par surface croissante (le plus léger en premier)
    best = sorted(solutions, key=lambda x: x['S'])[0]
    return best
```

Listing 4 – Fonction d'optimisation

Exemple du code et valeurs :
On cherche la section la plus légère pour une poutre de 4m devant supporter 100N.

```
print(optimisation_section(4, 100, E_ALU, NU_ALU))
OPTIMUM : e=1.0mm, b=1.0cm, h=1.0cm (Masse lin.=0.08 kg/m, F_crit=119.2N)
```

Le programme retourne les dimensions optimales (e, b, h) minimisant la masse linéique tout en respectant la contrainte de flambage.

3.5 Calcul de la masse de l'aile

Conformément au cours de Bilan de Masse, la formule de **F. Leclerc (2002)** a été implémentée. Le programme extrait automatiquement les valeurs $(MTOW, S, b, \varphi)$ du fichier de données 'Aircraft_Data.xlsx' pour effectuer ce calcul.

La formule utilisée est :

$$M_W = 0.197 \left(\frac{M_{MTOW} \cdot \lambda \cdot (\epsilon + 1)}{e_{req} \cdot \cos(\varphi_{25})} \right)^{0.6} S^{0.3} \tag{2}$$

```python
def calcul_masse_aile_leclerc(MTOW, S, b, epsilon, phi_25_rad, e_r):
    # Allongement
    if S <= 0: return 0
    lamb = b**2 / S

    # Sécurité cosinus
    cos_phi = np.cos(phi_25_rad)
    if abs(cos_phi) < 0.001: cos_phi = 0.001

    term_1 = (MTOW * lamb * (epsilon + 1)) / (e_r * cos_phi)

    # Calcul final (Masse Wing)
    Mw = 0.197 * (term_1**0.6) * (S**0.3)

    return Mw
```

Listing 5 – Fonction calcul_masse_aile_leclerc

Exemple du code et valeurs (Cas réel : A300 B4) :
Les données sont extraites automatiquement de la première ligne du fichier Excel.

```
# Données extraites : MTOW=157500 kg, S=260 m2, b=44.8 m, Phi=23 deg
# e_r estimé à 1.575 m (15% de la corde emplanture 10.5m)
Mw = calcul_masse_aile_leclerc(157500, 260, 44.8, 0, rad(23), 1.575)
```

— **Résultat obtenu** : $M_W \approx 3\,742$ kg.

4 Analyse Aérodynamique (XFLR5)

4.1 Justification des choix

Pour assurer une modélisation cohérente, les choix suivants ont été faits :

— **Profil NACA 4412** : Retenu car modérément cambré, adapté aux conditions ($V_\infty = 10$ m/s, $Re \approx 3 \times 10^6$).

— **Méthode** : Analyse 2D via XFoil pour générer les polaires (Re 1M à 4M), puis méthode des panneaux 3D (VLM) pour l'aile complète.

— **Géométrie** : Envergure 36 m, corde constante 4 m, pas de twist ni de dièdre. Cette simplification est volontaire pour isoler les effets fondamentaux sans complexifier l'interprétation géométrique.

— **Conditions** : Analyse à $V_\infty = 10$ m/s pour des incidences de -2°, 0°et 4°.

4.2 Résultats obtenus

Le tableau ci-dessous résume les coefficients obtenus via l'analyse VLM :

Alpha	**CL**	**CD**	**Cm**	**XCP (m)**	**ZCP (m)**
-2°	0.18848	0.01147	-0.18882	5.165	1.288
0°	0.37910	0.01521	-0.27108	3.911	1.222
4°	0.75736	0.03012	-0.44468	3.294	1.190

TABLE 2 – Résultats de l'analyse VLM ($V_\infty = 10$ m/s)

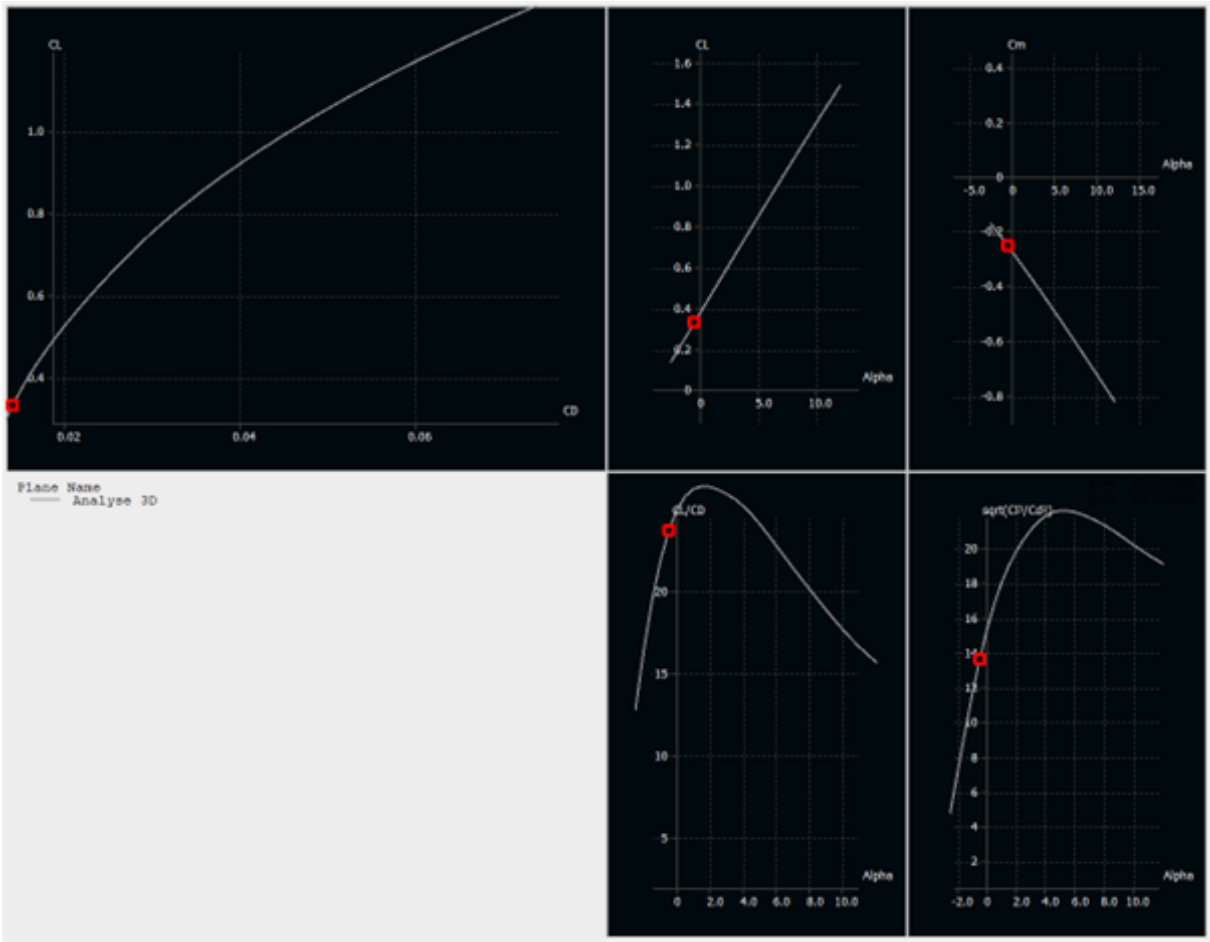

FIGURE 1 – Évolution des coefficients aérodynamiques (Polaire, C_L, C_m et Finesse)

4.3 Interprétation

1. **Portance (C_L)** :
   — À -2°, la portance est faible mais positive grâce à la cambrure.
   — À 0°, elle double presque.
   — À 4°, $C_L \approx 0.75$, indiquant un régime stable loin du décrochage.
2. **Traînée (C_D)** : Elle augmente logiquement avec l'incidence. Elle double à 4°(environ 0.03) par rapport au régime faible incidence, cohérent avec l'augmentation de la traînée induite ($\propto C_L^2$). L'optimum C_L/C_D se situe vers 2-3°.
3. **Moment de tangage (C_m)** : Toujours négatif (piqueur), comportement typique d'un profil cambré qui nécessitera un empennage horizontal compensateur.
4. **Centre de poussée (X_{CP})** : Il avance vers le bord d'attaque lorsque l'incidence augmente (de 5.16 m à 3.29 m).

4.4 Limites et Conclusion

La modélisation présente des limites connues : absence de fuselage (sous-estimation de la traînée), absence de twist (optimisation de charge alaire non réalisée), et limites de la méthode VLM sur le décrochage.

Néanmoins, l'analyse montre que l'aile équipée du profil NACA 4412 possède un comportement sain, performant et une marge de sécurité confortable. Les outils structurels et aérodynamiques sont désormais opérationnels pour la suite du projet.