# AER8270 - Aérodynamique

# Guide d'utilisation pour le code VLM

POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

# Vortex Lattice Method (VLM)

## Description

Le code VLM est un outil numérique qui résoud un écoulement potentiel en 3D, en d'autres mots, l'écoulement est non-visqueux, incompressible et irrotationel. L'équation à résoudre se simplifie à l'équation de Laplace

$$\nabla^2 \phi = 0 \tag{1}$$

qui est linéaire. Par conséquent, le théorême de superposition est valide, ce qui permet de représenter l'écoulement par plusieurs éléments vortex superposés.
L'approche VLM utilise des anneaux de vorticité (*vortex rings*) pour modéliser la surface de l'aile et générer une circulation (portance).
L'approche VLM peut donc être considérée comme une extension 3D de la théorie des profils minces anisi que de la ligne portante de Prandtl.
L'approche VLM permet bien modéliser une surface d'aile avec un effilement(*taper*), un angle de flèche(*sweep*) et une vrille(*twist*).

# Implémentation Python

## Fichiers

- *vlm.py* : Fichier principal à exécuter qui contient les fonctions pour construire et résoudre le système linéaire.
- *vortexRing.py* : Module qui définit une classe et les fonctions nécessaires pour modéliser des panneaux avec des vortex rings.
- *Vector3.py* : Module qui définit un objet de type vecteur (x,y,z).

## Attention

Tous les fichiers doivent être dans le même répertoire pour que le code fonctionne correctement.

# Utilisation

## Exemple #1

Vous pouvez exécuter directement dans un terminal le fichier *vlm.py*
***python vlm.py***
À la fin de ce-dernier, vous pouvez modifier les paramètres de la simulation.

```
if __name__ == '__main__':
    prob = VLM(ni=5,
               nj=20,
               chordRoot=1.0,
               chordTip=0.3,
               twistRoot=0.0,
               twistTip=0.0,
               span=5.0,
               sweep=0.0,
               Sref =3.250,
               referencePoint=[0.25,0.0,0.0],
               wingType=2,
               alphaRange = [0.0,5.0, 10.0, 20.0])
    prob.run()
```

run_example1.py

# Utilisation

## Exemple #2

Vous pouvez également créer votre propre script python et importer la *class* VLM du fichier *vlm.py*.

```
from vlm import *
prob = VLM(ni=5,
           nj=50,
           chordRoot=1.0,
           chordTip=1.0,
           twistRoot=0.0,
           twistTip=0.0,
           span=5.0,
           sweep=0.0,
           Sref =5.0,
           referencePoint=[0.0,0.0,0.0],
           wingType=1,
           alphaRange = [0.0,10.0])
prob.run()
```

run_example2.py

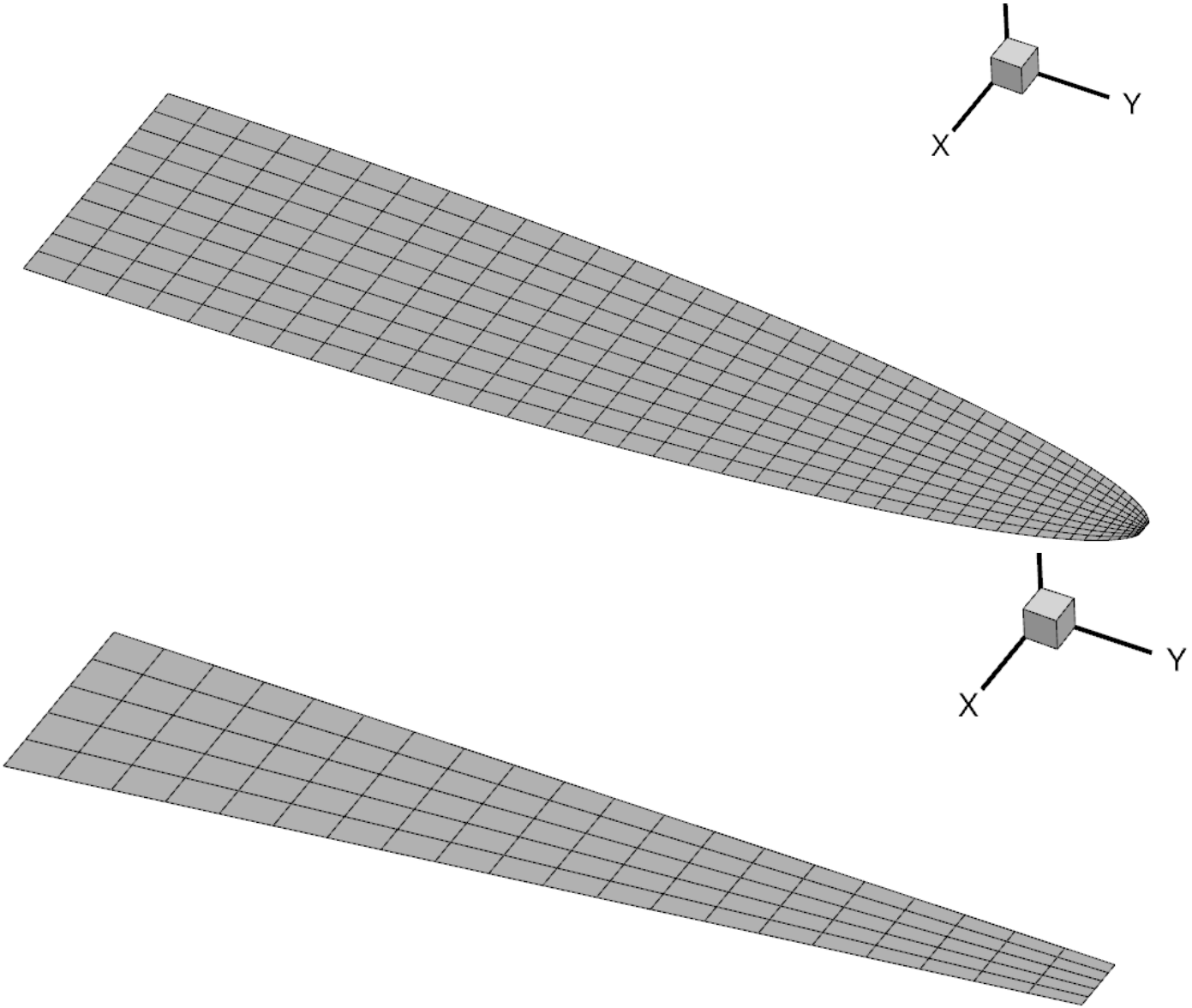
# Paramètres

Les paramètres d'entré sont pour pour une **demi aile**

## Class VLM : Input

- *ni* : Nombre de panneaux en x (corde)
- *nj* : Nombre de panneaux en y (envergure)
- *chordRoot* : Corde à l'emplanture [m]
- *chordTip* : Corde au bout de l'aile [m]
- *twistRoot* : Vrille à l'emplanture [deg]
- *twistTip* : Vrille au bout de l'aile [deg]
- *span* : Envergure de la demi-aile [m]
- *sweep* : Angle de flèche [deg]
- *Sref* : Surface de référence de la demi-aile [$m^2$]
- *referencePoint* : Point de référence pour le calcul du moment [m,m,m]
- *wingType* : Type d'aile considéré:
  1. 1: Aile régulière avec discrétisation y constante
  2. 2: Aile régulière avec discrétisation y cosinus
  3. 3: Aile Elliptique avec discrétisation y cosinus
- *alphaRange* : Liste d'angles d'attaques à simuler [deg]

# Représentation d'ailes



# Paramètres

## Class HSPM : Méthodes

- *run* : Exécute le code sur l'ensemble des angles d'attaques donnés en résolvant le système de panneaux créés et retourne un fichier *3D_sol_A...* et un fichier *Spanload_A...* pour chaque angle d'attaque. Le premier redonne les coordonnées XYZ des *nodes* des différents panneaux ainsi que $\Gamma$ au centre des panneaux. Le second donne le coefficient de portance local $C_l(y)$ en fonction de la position y (*spanload*)