TP2 Résolution des équations d'Euler isothermes compressibles en 2D avec la méthode des volumes finis

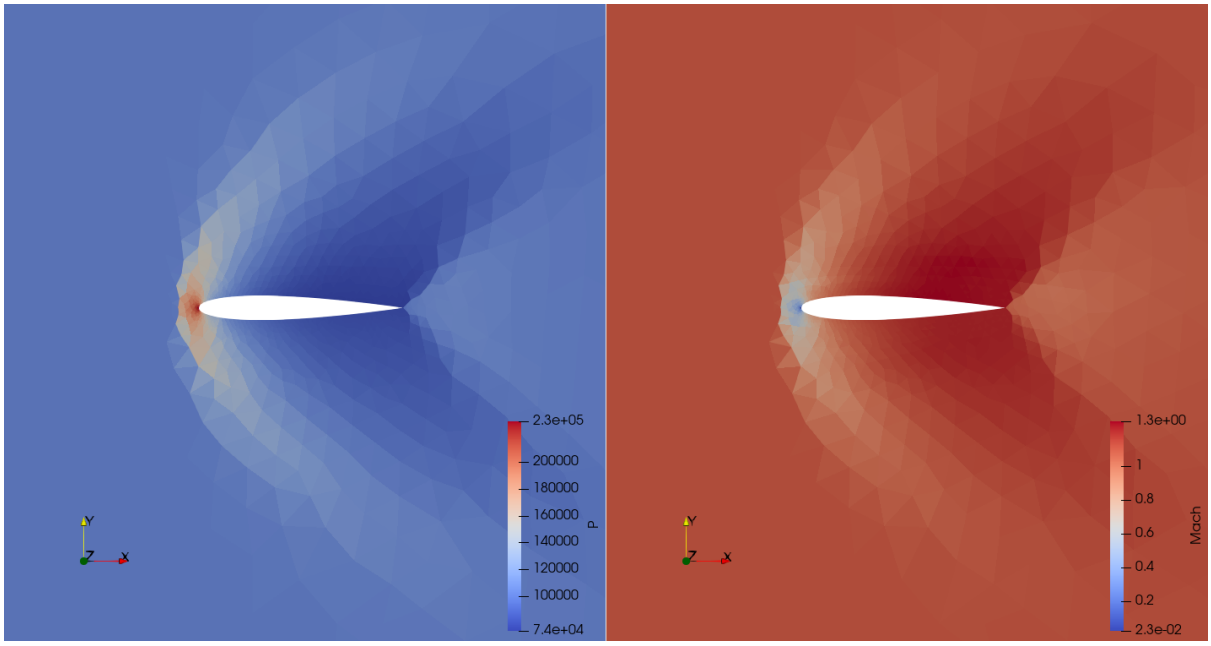

Figure 1: Champs de pression et de Mach autour d'un profil naca0012 pour $M_{inf} = 1.1$ obtenus avec le code du TP

Consignes:

- *Ce TP est noté et à rendre (au plus tard) pour le 19/12/2025;*
- *Le TP peut être fait en binôme;*
- *Le rendu du TP consiste en un rapport (succinct) ainsi que les deux fichiers main.py et solver.py;*
- *Une fois terminé, le TP est à envoyer à maxime.bouyges@onera.fr, compressé au format zip avec le nom suivant : TP2-NOM1_PRENOM1-NOM2_PRENOM2.* ***ATTENTION: le nom du fichier ne doit pas terminer par ".zip"****, sinon il sera filtré par ma messagerie. Il vous suffit de le renommer après la compression pour enlever le ".zip".*

Le but de ce TP est de construire un code volumes-finis non-structuré résolvant les équations d'Euler isothermes en 2D. La résolution de ces équations autour d'un profil

d'aile peut par exemple donner un ordre de la grandeur de la portance généré par ce profil en fonction de l'incidence, du Mach, etc.

Pour construire ce code, vous allez devoir compléter un code Python à trous. Ce code est divisé en différents fichiers selon une arborescence assez classique pour un code de résolution numérique. Chaque fonction possède une courte documentation, prenez le temps de bien la lire et de la comprendre. Seuls les fichiers `src/main.py` et `src/solver/solver.py` sont à modifier. Pour exécuter le code, il suffit de lancer la commande `python main.py`.

Dans un premier temps, pour simplifier le développement, le maillage utilisé est celui d'un rectangle avec seulement trois cellules suivant x et une suivant y.

1. **Énoncez** les équations d'Euler isothermes 2D (avec deux composantes de vitesse).

2. En écrivant ces équations sous la forme

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \frac{\partial \boldsymbol{F_x}(\boldsymbol{Q})}{\partial x} + \frac{\partial \boldsymbol{F_y}(\boldsymbol{Q})}{\partial y} = 0, \tag{1}$$

   détaillez le vecteur des variables conservatives ainsi que le vecteur flux associé à chaque dimension de l'espace.

3. Le flux de Roe pour ce système d'équation est déjà implémenté dans la fonction `solver/solver.py:roe_euler_isothermal_1D3`. Cette fonction fait appel au flux (analytique) des équations d'Euler demandé à la question 2.

   Complétez la fonction `solver/solver.py:euler_flux`.

4. Maintenant que nous disposons de fonctions permettant d'évaluer le flux numérique 1D, nous devons "tourner" ce flux pour le transformer en flux 2D. Pour cela, la fonction `solver/solver.py:compute_face_flux` est utilisée. Le principe est le suivant:

   - on applique une "rotation[1]" au vecteur q pour le ramener suivant la normale à la face;
   - on évalue le flux 1D avec cet état (`roe_euler_isothermal_1D3`);
   - on applique une "rotation" inverse au flux trouvé pour obtenir un flux 2D

   Avec cette fonction, le code est donc capable d'évaluer le flux numérique sur une face dans un espace en 2D.

   Complétez la fonction `solver/solver.py:compute_inner_flux` qui calcule le flux (2D) sur chaque face à l'intérieur du domaine.

5. (bonus) La question précédente a permis d'obtenir le flux sur les faces **internes**. Il reste à pouvoir calculer le flux sur les faces **limites**, c'est-à-dire appliquer les conditions limites. Ceci est réalisé par la fonction `solver.py:boundary_conditions` qui complète la "matrice" `flux` pour toutes les faces limites. **Quelle** est la condition à appliquer en présence d'une paroi? **Déterminez** l'état à appliquer dans la cellule fictive (ghost cell) pour appliquer la condition limite.

[1] Attention, ce ne sont pas des rotations au sens propre du terme, seules les grandeurs vectorielles (par exemple la vitesse) sont tournées, pas les scalaires comme ρ

6. **Complétez** la routine `solver/solver.py:compute_time_step` qui permet de calculer le pas de temps de la simulation en respectant un critère CFL. Pour le moment, on se contentera d'un pas de temps global, identique pour toutes les cellules (pour le moment, ignorez l'argument `local` de la fonction).

7. Il est temps de mettre toutes ces fonctions bout à bout pour résoudre un pas de temps. **Complétez** la fonction `solve_one_time_step`.

8. Enfin, **complétez** le corps de `main.py` pour effectuer une boucle en temps. **Lancez** le code, il doit tourner sans afficher d'erreur (éventuellement des avertissements).

9. **Changez** de fichier de maillage pour utiliser le fichier `mesh/naca0012.msh`. **Lancez** le calcul. À l'issue du calcul, deux fichiers de résultat sont créés:

   - un fichier `results.vtu`, contenant la solution dans tout le domaine, lisible par le logiciel de visualisation (gratuit) Paraview;
   - un fichier `surf.csv`, contenant l'extraction de la solution sur les parois du domaine.

10. **Utilisez** le fichier `surf.csv` pour tracer la pression à l'extrados et à l'intrados du profil. Idéalement, on tracera le coefficient de pression, c_p, autour du profil. **Commentez**.

11. (Questions bonus) Les questions sont indépendantes. Si vous répondez à l'une de ces questions, précisez-le dans votre rapport.

    (a) (facile) Pour une simulation stationnaire, il est possible d'adopter un pas de temps différent dans chaque cellule. On part de "pas de temps local". **Modifier** la routine `solver/solver.py:compute_time_step` pour utiliser un pas de temps local. **Quelles** conséquences voyez-vous pour la simulation (gain en temps de calcul, divergence numérique, physique mal représentée, autre)?

    (b) (facile) Pour suivre la convergence d'une simulation, on peut observer l'évolution de certaines grandeurs tels que les "résidus". La simulation effectuée ici étant stationnaire, on peut s'intéresser à l'écart entre la solution à l'instant n et la solution à l'instant $n-1$. **Tracez** l'évolution au cours du calcul de la norme de cet écart pour chaque variable. **Quand** peut-on supposer que la solution est convergée?

    (c) (intermédiaire) Le coefficient de portance global du profil s'exprime comme la valeur moyenne ($\sim$ intégrale) de l'écart des coefficients de pression entre intrados et extrados. **Tracez** l'évolution du coefficient de portance en fonction de l'incidence.

    (d) (difficile) Vous avez peut-être remarqué que certaines grandeurs liées au maillage sont calculées à chaque itération alors qu'elles sont constantes. Par exemple le volume de chaque cellule. La classe `Mesh` dispose d'un mécanisme permettant de pré-calculer certaines valeurs et de les stocker dans un cache. **Faites** appel à ce mécanisme pour accélérer les temps de calcul. **Mesurez** le gain en temps de calcul associé à l'utilisation de ce cache

    (e) (difficile) Le code utilise une intégration temporelle explicite, qui nécessite d'utiliser un CFL inférieur à un. **Mettez** en place une intégration temporelle implicite.

(f) (très difficile) Le code utilise une discrétisation spatiale et temporelle d'ordre un. **Appliquez** une discrétisation spatiale et temporelle d'ordre deux. Pour simplifier, on pourra supposer que toutes les mailles sont des rectangles.