Unterschrift BetreuerIn

BACHELORARBEIT

Implementation eines Hess-Smith Panelverfahrens in Python

ausgeführt am Institut für Strömungsmechanik und Wärmeübertragung
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Braun

durch

Leon Schwarzäugl

Gumpendorfer Straße 137,
1060 Wien

29. Juli 2022

Unterschrift StudentIn

Zusammenfassung

Im Zuge der vorliegenden Arbeit wurde ein Panelverfahrens nach Hess-Smith zur Berechnung von ebenen Potentialströmungen um geschlossene Körper implementiert und auf verschiedene Probleme angewandt. Dazu wurde die Programmiersprache Python verwendet.
Zunächst wurde eine Klasse für Panele und ihre Eigenschaften entworfen. Im Zuge dessen wurde eine Routine entwickelt, die jene Parameter in eine .csv-Datei ausgibt. Danach wurde eine Klasse für Tragflächenprofile sowie Methoden zur Berechnung der ihnen zugehörigen Systemparameter implementiert. Mit den somit berechneten Systemparametern kann das sich ergebende lineare Gleichungssystem für die Quellstärken gelöst werden; im Anschluss wurden aus den ermittelten Quellstärken die Tangentialgeschwindigkeiten und Druckbeiwerte an den jeweiligen Panelen, sowie unter Hinzunahme einer Wirbelbewegung der Auftriebsbeiwert für das Gesamtsystem ermittelt. Abschließend wurde eine Routine für die Ermittlung der Tangentialgeschwindigkeiten an beliebigen Punkten auf der Oberfläche des Profils entwickelt.
Die fertige Implementation wurde zunächst verwendet, um für die Panele verschiedener Tragflächenprofile Graphen zu Geometrien und Parametern zu gewinnen. Anschließend wurde anhand eines Kreiszylinders mit 8 Panelen gleicher Seitenlänge der Vektor der Quellbewegung, sowie die Tangentialgeschwindigkeiten und Druckbeiwerte an den Panelmittelpunkten ermittelt sowie die Ergebnisse für den Druckbeiwert mit dem theoretischen Ergebnis aus der Potentialtheorie verglichen. Dabei wurde eine gute Übereinstimmung ermittelt. Abschließend wurde das oben genannte System um eine Wirbelbewegung erweitert und der Auftriebsbeiwert unter verschiedenen Anstellwinkeln des Profils bei konstantem Angriffswinkel berechnet.
Letztlich wurde die Implementierung auf Joukowski-Profile angewandt und eine experimentelle Abschätzung der Fehlerordnung des Hess-Smith-Verfahrens durchgeführt.

Inhaltsverzeichnis

1 Aufgabenstellung

Die Berechnung der aerodynamischen Eigenschaften eines Tragflächenprofils.

1.1 Zielsetzung

Es sollen folgende Punkte implementiert werden:

- Eine Klasse für Panele und die sie definierenden Eigenschaften: Mittelpunkte auf der x- und y-Achse, Neigungswinkel θ_i und Länge l_i. Jedes Panel enthält ebenfalls Möglichkeiten zur Speicherung der zugehörigen Quellstärken q_i, Tangentialgeschwindigkeiten $v_i^{(t)}$ und Druckbeiwerten c_{p_i}
- Eine Klasse für Trägerprofile, ihre Umfänge U und Tiefe t sowie mathematisch relevante Systemparameter ξ_{ij}, η_{ij}, I_{ij}, J_{ij}, $A_{ij}^{(n)}$, $A_{ij}^{(t)}$, M_{ij}.
- Eine Methode zur Lösung des linearen Gleichungssystems $M\vec{q} = \vec{b}$ für den Vektor der Quellbewegung unter einer konstanten Anströmgeschwindigkeit V_∞ und einem Anstellwinkel des Profils α, sowie der Berechnung der daraus resultierenden Tangentialgeschwindigkeiten und Druckbeiwerte.

Weiters soll mittels der Implementation eine Untersuchung der Abweichung der ermittelten Druckbeiwerte von den aus der Potentialtheorie berechneten Werten für einen achtseitigen Kreiszylinder erfolgen.

1.2 Methodik

Zur Lösung der Aufgabenstellung wurde die Programmiersprache Python verwendet. Für die Lösung der linearen Gleichungssysteme wird linalg.solve[1] aus der Python-Bibliothek NumPy verwendet.
Sämtliche Graphen wurden mithilfe der Python-Bibliothek Matplotlib erstellt. [5]
Sofern analytische Lösungen zu den linearen Gleichungssystemen berechnet wurden, wurde dafür die Python-Bibliothek SciPy verwendet. [6]

[1] linalg.solve verwendet die LAPACK routine _gesv als Solver [3]

2 Grundlagen

2.1 Hess-Smith-Panelverfahren

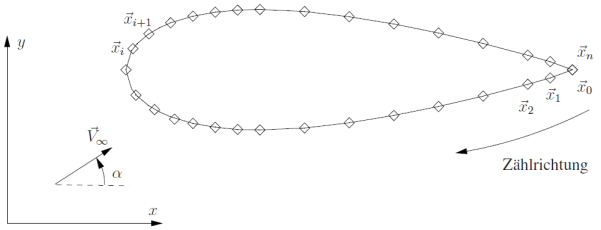


Abbildung 2.1: Zur Definition der Zählrichtung der Panels sowie der Anströmgeschwindigkeit und deren Winkel

Das Hess-Smith-Panelverfahren ist ein Verfahren zur Berechnung von ebenen Potentialströmungen um geschlossene Körper mit Auftrieb. Ein kontinuierlicher zweidimensionaler Körper mit der Umfangkurve $\mathcal{C}$ wird dabei zunächst durch $n+1$ diskrete Datenpunkte $\vec{x}_i = (x_i, y_i)$ dargestellt. Die Zählrichtung dieser Punkte sei als im Uhrzeigersinn festgelegt (siehe Abb. 2.1). Für das Profil gilt somit die Periodizität $\vec{x}_n = \vec{x}_0$.
Die Kontur des Profils wird nun durch eine endliche Anzahl Verbindungsgeraden zwischen zwei benachbarten Punkten $\vec{x}_i, \vec{x}_{i+1}$, den Panelen $\mathcal{C}_i$ (Abb. 2.2) angenähert. Dabei wird jedes Panel durch die folgenden Parameter charakterisiert:

- Den Mittelpunkten auf beiden Achsen:

$$X_i = \frac{x_{n-i} + x_{n-i-1}}{2},\ Y_i = \frac{y_{n-i} + y_{n-i-1}}{2}, \tag{2.1}$$

- seinem Neigungswinkel:

$$\theta_i = \left(\frac{y_{n-i-1} - y_{n-i}}{x_{n-i-1} - x_{n-i}} \right), \tag{2.2}$$

- und seiner Länge:

$$l_i = \sqrt{(x_{n-i-1} - x_{n-i})^2 + (y_{n-i-1} - y_{n-i})^2}. \tag{2.3}$$

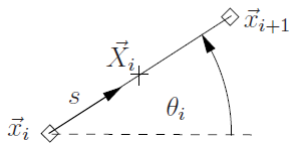


Abbildung 2.2: Zur Definition eines Panels

Im Hess-Smith-Panelverfahren sei nun jedes dieser Panels mit einer vorerst unbestimmten, konstanten Quelldichte q_i behaftet, welche an einem Punkt $\vec{r}_i = \vec{x} - \hat{x}_i(s)$ zum Geschwindigkeitspotential den Beitrag

$$\phi_i^{(q)} = \frac{q_i}{2\pi} \int_{\mathcal{C}_i} ds \ln r_i \tag{2.4}$$

liefert. s ist dabei die Koordinate der Bogenlänge, welche das gesamte Profil parametrisiert.
Ebenso liefert die Profilanströmung mit der konstanten Geschwindigkeit V_∞ unter dem Winkel α ein Potential

$$\phi_\infty = V_\infty(\cos\alpha x + \sin\alpha y). \tag{2.5}$$

Aus der Summe über alle Panelbeiträge aus (2.4) und (2.5) ergibt sich das Gesamtgeschwindigkeitspotential zu

$$\phi(\vec{x}) = V_\infty(\cos\alpha x + \sin\alpha y) + \sum_{i=0}^{n-1} \phi_i^{(q)}. \tag{2.6}$$

Daraus folgt für die Geschwindigkeit im Punkte $\vec{x}$

$$\vec{v}(\vec{x}) = \nabla\phi(\vec{x}). \tag{2.7}$$

Auf der Oberfläche des Profils muss die Strömung die Gleitbedingung erfüllen; damit lassen sich n Gleichungen aufstellen, um diese Bedingung zu modellieren. Wir verlangen, dass die Normalkomponente des Geschwindigkeitsvektors,

$$v_j^{(n)} = \vec{v}(\vec{X}_j) \cdot \vec{n}_j, \tag{2.8}$$

verschwindet. Dabei ist $\vec{n}_j$ der Normalvektor auf das Panel $\mathcal{C}_j$.
Wir erhalten in 0. Ordnung ein System von n Gleichungen

$$\sum_{j=0}^{n-1} M_{ij}^{(q)} q_j = b_i, \tag{2.9}$$

mit

$$M_{ij}^{(q)} = \frac{1}{q_j} \nabla \phi_j^{(q)}(\vec{X}) \cdot \vec{n}_i, \tag{2.10}$$

$$b_i = -V_\infty \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \cdot \vec{n}_i, \tag{2.11}$$

aus dessen Lösung wir die Quellstärken q_i für jedes Panel ermitteln können.
Damit kann nun der Tangentialkomponenten des Geschwindigkeitsvektors

$$v_j^{(t)} = \vec{v}(X_j) \cdot \frac{\vec{x}_{j+1} - \vec{x}_j}{|\vec{x}_{j+1} - \vec{x}_j|}, \tag{2.12}$$

bestimmt werden.
Aus der Bernoulli-Gleichung

$$p_\infty + \frac{1}{2}\rho V_\infty^2 = p + \frac{1}{2}\rho v^2, \tag{2.13}$$

definieren wir den Druckbeiwert als das Verhältnis zwischen der Druckdifferenze zwischen der Profilanströmung und dem dynamischen Druck

$$c_p = \frac{p - p_\infty}{\frac{1}{2}\rho V_\infty^2}. \tag{2.14}$$

Daraus können wir den Druckbeiwert

$$c_{p_j} = 1 - \left(\frac{v_j^{(t)}}{V_{\infty,}} \right) \tag{2.15}$$

am Mittelpunkt des Panels $\mathcal{C}_j$ bestimmen.

2.1.1 Berücksichtigung einer Wirbelbewegung

Wollen wir den Auftrieb berücksichtigen, so reicht die reine Quellbelegung des Profils noch nicht aus.
Im Hess-Smith Verfahren nehmen wir für das gesamte Profil eine einzige konstante Wirbelbelegung γ für alle Panels an. Damit wird das Potential (2.6) erweitert um Beiträge

$$\phi_i^{(w)} = \frac{\gamma}{2\pi} \int ds \theta_i, \tag{2.16}$$

zu

$$\begin{aligned} \phi(\vec{x}) &= V_\infty(\cos\alpha x + \sin\alpha y) + \sum_{i=0}^{n-1} \left(\phi_i^{(q)} + \phi_i^{(w)} \right) \\ &= V_\infty(\cos\alpha x + \sin\alpha y) + \sum_{i=0}^{n-1} \int_{\mathcal{C}_i} \left(\frac{q_i}{2\pi} \ln r_i - \frac{\gamma}{2\pi} \theta_i \right) ds \end{aligned} \tag{2.17}$$

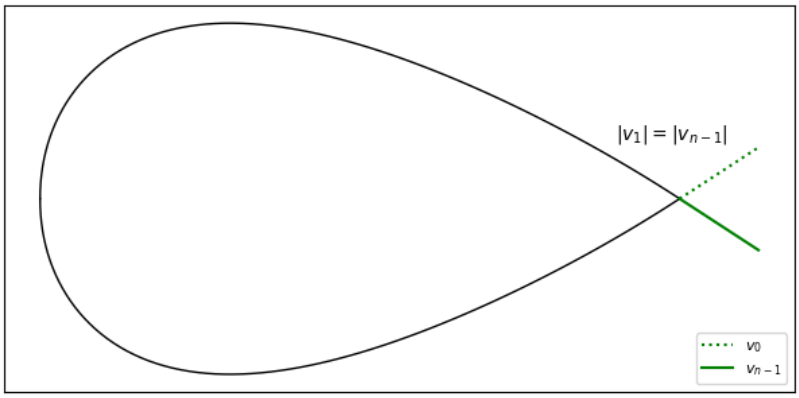


Abbildung 2.3: Zur Kutta-Bedingung am Beispiel eines NACA0012-Profils - am Punkt der Hinterkante sind die Tangentialgeschwindigkeiten genau entgegengerichtet

Die Gleitbedingung bleibt erhalten, allerdings muss das Gleichungssystem aber entsprechend dem hinzugekommenen Geschwindigkeitsbeitrag erweitert werden. Für die neue Unbekannte γ fehlt also eine Gleichung. Diese ergibt sich aus der Kutta-Bedingung, welche besagt, dass es an der Hinterkante des Profils keine Umströmung gibt.
Zur Modellierung der Kutta-Bedingung wählen wir

$$v_0^{(t)} = -v_n^{(t)}. \tag{2.18}$$

Daraus folgt, dass die Tangentialkomponenten des Geschwindigkeitsvektors in den Mittelpunkten der Panels, welche die Profilhinterkante bilden, vom Betrag gleich groß und gegenläufig gerichtet sein muss (siehe Abb. 2.3).
Nach Lösung des erhaltenen linearen Gleichungssystems mit $q_n = \gamma$ können wir nun den Auftriebsbeiwert c_a ermitteln. Dieser ergibt sich mit t, der Proftiltiefe und l_i, der Länge des Panels $\mathcal{C}_i$, approximiert in folgender Form:

$$c_a \approx \frac{2}{V_{\infty t}} \sum_{i=0}^{n-1} v_i^{(t)} l_i. \tag{2.19}$$

Für eine analytische Berechnung des Auftriebsbeiwerts müssen wir nun noch die Systemmatrix bestimmen, welche das lineare Gleichungssystem ((2.9)) definiert. [4] [2] [1]

2.1.2 Berechnung der Systemmatrix

Die Systemmatrix zur Berechnung der Normalgeschwindigkeiten $v_i^{(n)}$ in den Panelmittelpunkten für eine stückweise konstante Quellenbewegung der Panels ergibt sich aus der Gleitbedingung. Wenden wir diese direkt auf 2.6 an, muss ein kompliziertes Integral gelöst werden. Dieses kann theoretisch direkt mit Computermethoden gelöst werden, für den Berechnungsaufwand ist es allerdings gerade bei Profilen mit einer hohen Anzahl an Panels günstiger, dieses analytisch zu berechnen.
Für die Systemmatrix und ihre Parameter gelten:

$$M_{ij} = A_{ij}^{(n)}, \quad i = 0, 1, \ldots, n-1; \quad j = 0, 1, \ldots, n-1. \tag{2.20}$$

mit der Einflussmatrix der Quellbelegung

$$A_{ij}^{(n)} = -\sin(\theta_i - \theta_j) I_{ij} + \cos(\theta_i - \theta_j) J_{ij} \tag{2.21}$$

sowie das geometrische Integral der Normalgeschwindigkeit

$$I_{ij} = \begin{cases} \frac{1}{4\pi} \ln\left[\frac{(l_j + 2\xi_{ij})^2 + 4\eta_{ij}^2}{(l_j - 2\xi_{ij})^2 + 4\eta_{ij}^2}\right] & i \neq j \\ 0 & i = j \end{cases} \tag{2.22}$$

und dem geometrischen Integral der Tangentialgeschindigkeit

$$J_{ij} = \begin{cases} \frac{1}{2\pi} \arctan\left[\frac{l_j - 2\xi_{ij}}{2\eta_{ij}}\right] + \frac{1}{2\pi} \arctan\left[\frac{l_j + 2\xi_{ij}}{2\eta_{ij}}\right] & i \neq j \\ \frac{1}{2} & i = j \end{cases} \tag{2.23}$$

$$\xi_{ij} = (X_i - X_j)\cos\theta_j + (Y_i - Y_j)\sin\theta_j \tag{2.24}$$

$$\eta_{ij} = -(X_i - X_j)\sin\theta_j + (Y_i - Y_j)\cos\theta_j \tag{2.25}$$

Mit der Systemmatrix und den Inhomogenitäten b_i können wir nun das lineare Gleichungssystem aus (2.9) berechnen.
Für die Tangentialkomponente der Geschwindigkeit erhalten wir dann den Ausdruck

$$v_i^{(t)} = \sum_{j=0}^{n-1} A_{ij}^{(t)} q_j + V_\infty \cos(\alpha - \theta_i), \tag{2.26}$$

mit der Einflussmatrix der Wirbelbelegung

$$A_{ij}^{(t)} = \cos(\theta_i - \theta_j) I_{ij} + \sin(\theta_i - \theta_j) J_{ij}. \tag{2.27}$$

Diese Systemmatrix berücksichtigt allerdings noch keine Wirbelbelegungen. Im

Falle einer konstanten Wirbelbelegung γ auf allen Panels erweitert sich die Dimension von M_{ij} von $n \times n$ auf $(n+1) \times (n+1)$. Dadurch müssen folgende zusätzliche Elemente bestimmt werden:

$$M_{in} = \sum_{j=0}^{n-1} A_{ij}^{(t)}, \quad i = 0, 1, \ldots, n-1; \tag{2.28}$$

$$M_{nj} = A_{0j}^{(t)} + A_{n-1,j}^{(t)}, \quad j = 0, 1, \ldots, n-1; \tag{2.29}$$

$$M_{n,n} = -\sum_{j=0}^{n-1} \left[A_{0,j}^{(n)} + A_{n-1,j}^{(n)} \right]; \tag{2.30}$$

$$b_n = -V_\infty [\cos(\alpha - \theta 0) + \cos(\alpha - \theta_{n-1})]; \tag{2.31}$$

Gleichung (2.26) erweitert sich damit zu

$$v_i^{(t)} = \sum_{j=0}^{n-1} A_{ij}^{(t)} q_j - \gamma \sum_{j=0}^{n-1} A_{i,j}^{(n)} + V_\infty \cos(\alpha - \theta_i). \tag{2.32}$$

3 Bericht der Arbeit

Im Folgenden werden die wesentlichen Erkenntnisse der Arbeit präsentiert.

3.1 Vorbereitung der Daten

Die Daten der Profile wurden der UIUC Airfoil Data Site[1] entnommen. Die Daten lagen dabei überwiegend im Selig- oder Lednicer-Format vor.[2]
Im Lednicer-Format wird in der ersten Zeile die Bezeichnung des Profils angegeben. In der zweiten Zeile wird die Anzahl der Koordinatenpaare der Ober- und Unterseite angegeben. Ab der dritten Zeile werden die Koordinaten der Panelenden (x_i, y_i) von $x = 0$ bis $x = 1$ für die Oberseite des Profils angegeben. Danach folgt eine Leerzeile. Danach werden die Koordinaten der Panelenden (x_i, y_i) von $x = 0$ bis $x = 1$ für die Unterseite des Profils angegeben.
Im Selig-Format wird in der ersten Zeile die Bezeichnung des Profils angegeben. Ab der zweiten Zeile werden die Koordinaten der Panelenden (x_i, y_i) im Gegenuhrzeigersinn, beginnend bei $x = 1$ angegeben.

Es wurde eine Routine geschrieben, welche Daten im Lednicer-Format in das Selig-Format überführt. Dies wurde bewerkstelligt, indem die Koordinatenpaare der Oberseite invertiert wurden. Anschließend wurden aus den Daten die Header-Zeilen entfernt.
Unter Verwendung der Methode .write_panels() wurde für sämtliche Profile eine .csv-Datei mit den Werten X_i, Y_i, θ_i, l_i pro Panel erstellt (eine Beispieldatei ist in Anhang A: Lednicer- und Selig-Format gezeigt. Ebenso wurde für jedes sih in der UIUC-Datenbank befindliche Trägerprofil ein Graph erzeugt, welcher zum einen die Geometrie des Profils, sowie auch die Neigungswinkel und Längen der einzelnen Panels darstellt. Zur einfacheren Betrachtung sind die Graphen der Neigungswinkel und Panellängen in die beiden Seiten des Profils aufgeteilt.
Ein solcher Graph ist in Abbildung 3.1 an einem NACA-0012-Profil gezeigt. Die Symmetrie des Profils spiegelt sich in der gewählten Darstellung der Neigungswinkel und Panellängen gut wieder.

[1] `https://m-selig.ae.illinois.edu/ads/coord_database.html`

[2] Eine Gegenüberstellung der beiden Formate ist in Anhang A: Lednicer- und Selig-Format zu finden.

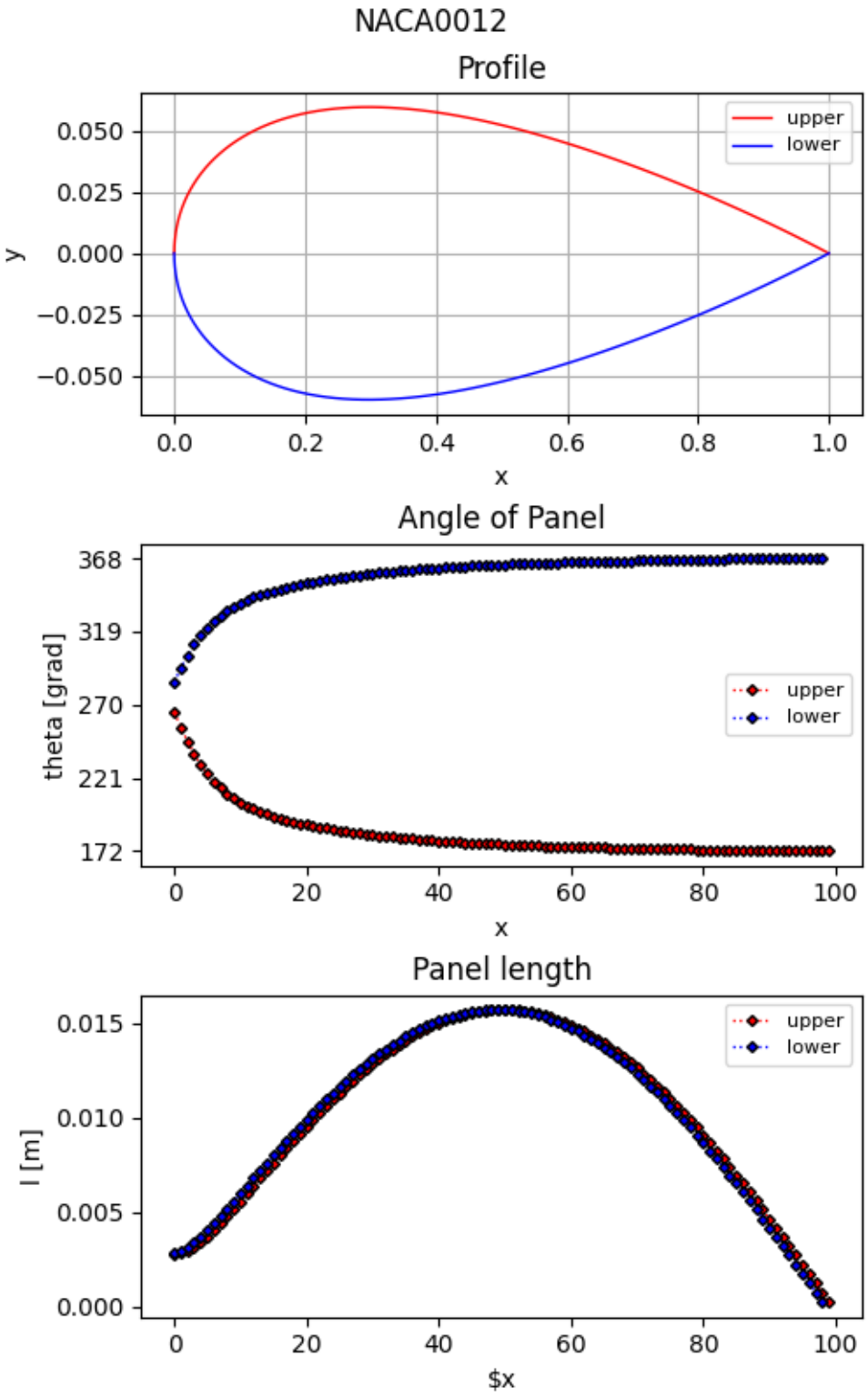


Abbildung 3.1: NACA-0012-Profil

X_i	Y_i	θ_i	l_i
0.8	-0.35	67.5	0.77
0.35	-0.85	22.5	0.77
-0.35	-0.85	337.5	0.77
-0.85	-0.35	292.5	0.77
-0.85	0.35	247.5	0.77
-0.35	0.85	202.5	0.77
0.35	0.85	157.5	0.77
0.85	0.35	112.5	0.77

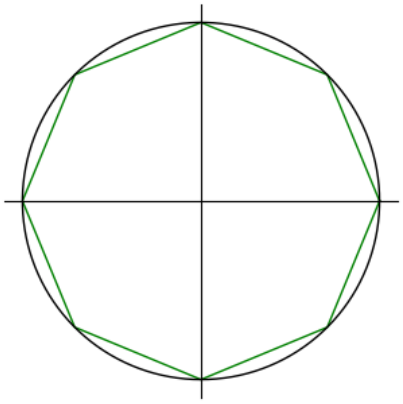


Abbildung 3.2: Achtseitiger Cylinder als Approximation eines Kreiszylinders sowie zugehörige Panelparameter

3.2 Untersuchungen am Kreiszylinder

Als nächstes wurden Untersuchung an Kreiszylindern vorgenommen, da bei diesen die Aufteilung in Panele relativ leicht zu implementieren ist. Zur Generierung eines Kreiszylinders mit n gleich großen Seiten wurde die Funktion make_cylinder(r, n) verwendet (siehe Anhang C: Codeausschnitte).

3.2.1 Achtseitiger Kreiszylinder

Berechnung der Systemmatrix

Für den in 3.2 gezeigten Kreisylinder wurde die Systemmatrix, gerundet auf zwei Dezimalstellen, zu folgenden Werten berechnet:

$$M_{i,j} = \begin{pmatrix} 0.5 & 0.056 & 0.064 & 0.065 & 0.065 & 0.065 & 0.064 & 0.056 \\ 0.056 & 0.5 & 0.056 & 0.064 & 0.065 & 0.065 & 0.065 & 0.064 \\ 0.064 & 0.056 & 0.5 & 0.056 & 0.064 & 0.065 & 0.065 & 0.065 \\ 0.065 & 0.064 & 0.056 & 0.5 & 0.056 & 0.064 & 0.065 & 0.065 \\ 0.065 & 0.065 & 0.064 & 0.056 & 0.5 & 0.056 & 0.064 & 0.065 \\ 0.065 & 0.065 & 0.065 & 0.064 & 0.056 & 0.5 & 0.056 & 0.064 \\ 0.064 & 0.065 & 0.065 & 0.065 & 0.064 & 0.056 & 0.5 & 0.056 \\ 0.056 & 0.064 & 0.065 & 0.065 & 0.065 & 0.064 & 0.056 & 0.5 \end{pmatrix}$$

Lösung des wirbellosen Gleichungssystems

Im Folgenden wurde die Lösung des linearen Gleichungssystems mit der obigen Systemmatrix und den Inhomogenitäten

$$b_i = -V_\infty \sin(q - \theta_i), \quad i = 0, 1, \ldots, n-1 \tag{3.1}$$

berechnet. Daraus wurde der Vektor der Quellenbewegung zunächst mit V_∞ und α als Variablen berechnet:

$$\vec{q} = \begin{pmatrix} -0.16V_\infty \sin(\alpha - 5.89) - 0.17V_\infty \sin(\alpha - 5.11) - 0.17V_\infty \sin(\alpha - 4.32) - 0.17V_\infty \sin(\alpha - 3.53) - 0.16V_\infty \sin(\alpha - 2.75) - 0.12V_\infty \sin(\alpha - 1.96) + 2.13V_\infty \sin(\alpha - 1.18) - 0.12V_\infty \sin(\alpha - 0.39) \\ -0.12V_\infty \sin(\alpha - 5.89) - 0.16V_\infty \sin(\alpha - 5.11) - 0.17V_\infty \sin(\alpha - 4.32) - 0.17V_\infty \sin(\alpha - 3.53) - 0.17V_\infty \sin(\alpha - 2.75) - 0.16V_\infty \sin(\alpha - 1.96) - 0.12V_\infty \sin(\alpha - 1.18) + 2.13V_\infty \sin(\alpha - 0.39) \\ 2.13V_\infty \sin(\alpha - 5.89) - 0.12V_\infty \sin(\alpha - 5.11) - 0.16V_\infty \sin(\alpha - 4.32) - 0.17V_\infty \sin(\alpha - 3.53) - 0.17V_\infty \sin(\alpha - 2.75) - 0.17V_\infty \sin(\alpha - 1.96) - 0.16V_\infty \sin(\alpha - 1.18) - 0.12V_\infty \sin(\alpha - 0.39) \\ -0.12V_\infty \sin(\alpha - 5.89) + 2.13V_\infty \sin(\alpha - 5.11) - 0.12V_\infty \sin(\alpha - 4.32) - 0.16V_\infty \sin(\alpha - 3.53) - 0.17V_\infty \sin(\alpha - 2.75) - 0.17V_\infty \sin(\alpha - 1.96) - 0.17V_\infty \sin(\alpha - 1.18) - 0.16V_\infty \sin(\alpha - 0.39) \\ -0.16V_\infty \sin(\alpha - 5.89) - 0.12V_\infty \sin(\alpha - 5.11) + 2.13V_\infty \sin(\alpha - 4.32) - 0.12V_\infty \sin(\alpha - 3.53) - 0.16V_\infty \sin(\alpha - 2.75) - 0.17V_\infty \sin(\alpha - 1.96) - 0.17V_\infty \sin(\alpha - 1.18) - 0.17V_\infty \sin(\alpha - 0.39) \\ -0.17V_\infty \sin(\alpha - 5.89) - 0.16V_\infty \sin(\alpha - 5.11) - 0.12V_\infty \sin(\alpha - 4.32) + 2.13V_\infty \sin(\alpha - 3.53) - 0.12V_\infty \sin(\alpha - 2.75) - 0.16V_\infty \sin(\alpha - 1.96) - 0.17V_\infty \sin(\alpha - 1.18) - 0.17V_\infty \sin(\alpha - 0.39) \\ -0.17V_\infty \sin(\alpha - 5.89) - 0.17V_\infty \sin(\alpha - 5.11) - 0.16V_\infty \sin(\alpha - 4.32) - 0.12V_\infty \sin(\alpha - 3.53) + 2.13V_\infty \sin(\alpha - 2.75) - 0.12V_\infty \sin(\alpha - 1.96) - 0.16V_\infty \sin(\alpha - 1.18) - 0.17V_\infty \sin(\alpha - 0.39) \\ -0.17V_\infty \sin(\alpha - 5.89) - 0.17V_\infty \sin(\alpha - 5.11) - 0.17V_\infty \sin(\alpha - 4.32) - 0.16V_\infty \sin(\alpha - 3.53) - 0.12V_\infty \sin(\alpha - 2.75) + 2.13V_\infty \sin(\alpha - 1.96) - 0.12V_\infty \sin(\alpha - 1.18) - 0.16V_\infty \sin(\alpha - 0.39) \end{pmatrix}$$

Nach Einsetzen der Werte $V_\infty = 1$ und $\alpha = 0$ vereinfacht sich der Vektor zu einem leichter zu handhabenden Wert. Die Ergebnisse für Quellenbelegung, Tangentialgeschwindigkeiten und Druckbeiwerte sind in Tabelle 3.1 zusammengetragen. Ebenfalls von Interesse ist die Größe $\sum_{i=0}^{n-1} q_i l_i$. Diese Summe sollte für einen geschlossenen Körper null ergeben, da der Körper sonst durch den Fluss seine Masse verändern würde. Tatsächlich ergibt sich:

$$\sum_{i=0}^{n-1} q_i l_i \approx 4.44 \cdot 10^{-16}$$

Dieser sehr kleine Fehler ergibt sich zum einen aus Fehlern, die bei der Berechnung von $\vec{q}$ entstehen, und zum anderen aus Rundungsfehlern bei der Berechnung der Eckpunkte des Kreiszylinders.

Tabelle 3.1: Ergebnisse für den achtseitigen Zylinder mit $V_\infty = 1, \alpha = 0$

i	q_i	$v_i^{(t)}$	c_{p_i}
0	-2.19	0.77	0.41
1	-0.91	1.85	-2.41
2	0.91	1.85	-2.41
3	2.19	0.77	0.41
4	2.19	-0.77	0.41
5	0.91	-1.85	-2.41
6	-0.91	-1.85	-2.41
7	-2.19	-0.765	0.41

3.2.2 Untersuchungen an variierenden Kreiszylindern

Die Potentialtheorie liefert ein exaktes Resultat für den Druckbeiwert auf der Oberfläche eines Zylinders:

$$c_p^{\text{exakt}}(\varphi) = 2\cos(2 * \varphi) - 1 \tag{3.2}$$

Dabei bezeichnet φ den Winkel der Flächennormale des Kreiszylinders welche ins Innere des Profils zeigt. Mit unserer Definition des Neigungswinkels θ gilt

$$\varphi = \theta + \frac{\pi}{2}. \tag{3.3}$$

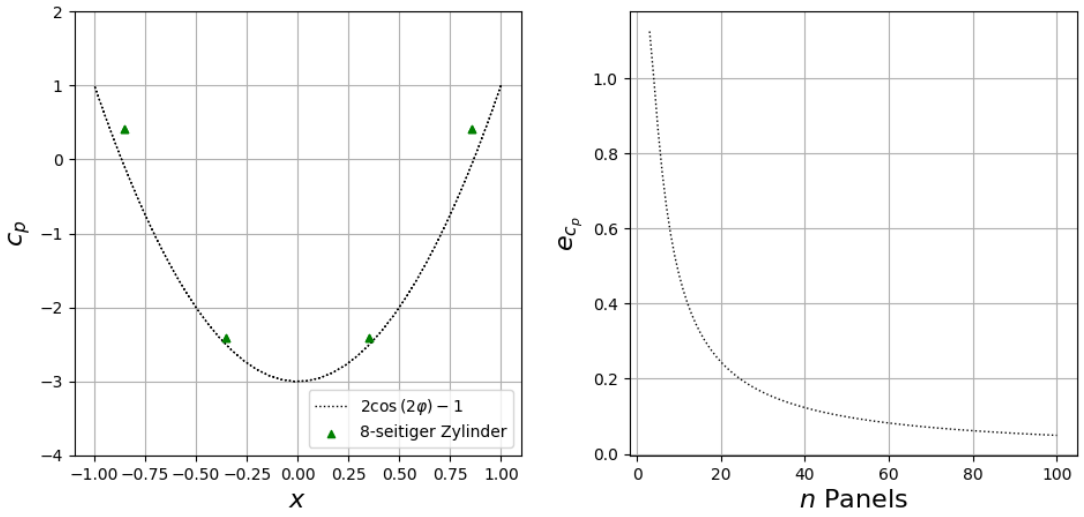
Abbildung **??** zeigt die Abweichung der Druckbeiwerte eines achtseitigen Zylinders vom analytischen Resultat. Es ergibt sich eine Abweichung dadurch, dass die Mittelpunkte des achtseitigen Zylinders nicht auf der Kreisfläche liegen. Wir können den Fehler durch eine Umformung bestimmen:

$$\begin{aligned} c_p^{\text{exakt}}(\varphi) &= 2\cos(2\varphi) - 1 && (3.4)\\ &= 1 - 4\sin^2\varphi && (3.5)\\ &= 1 - 4\left(\frac{Y}{R}\right)^2 && (3.6) \end{aligned}$$

Der Fehler, der bei der Approximation entsteht, ist also

$$\begin{aligned} \Delta c_{p_i} &= |c_{p_i} - c_p^{\text{exakt}}(\varphi)| && (3.7)\\ &= |c_{p_i} - [1 - 4(Y_i/R)^2]\,| && (3.8) \end{aligned}$$

Das Programm wurde nun für Kreiszylinder zwischen 5 und 100 Panels durchlaufen und die mittlere absolute Abweichung e_{c_p} für jeden Kreiszylinder festgehalten (Abbildung 3.3b). Das Experiment zeigt, dass bereits ab einer geringen Anzahl an Panels eine akzeptable Approximation erreicht wird.



(a) (b)

Abbildung 3.3: a) Abweichung der Druckbeiwerte eines achtseitigen Zylinders vom theoretisch ermittelten Wert b) Graph der Abweichung des Relativen Fehlers vom exaken Wert.

Variation der Panelanzahl

Rotation des Zylinders

3.3 Untersuchungen an ausgewählten Profilen

3.3.1 NACA0012-Profil

3.3.2 Joukowsky 12%-Profil

3.3.3 Fehlerabschätzung des Hess-Smith-Verfahrens

Literatur

[1] J.J. Alonso. *Hess-Smith Panel Method*. 2005. URL: http://aero-comlab.stanford.edu/aa200b/lect_notes/lect3-4.pdf (besucht am 28.07.2022).

[2] T. Cebeci. *An Engineering Approach to the Calculation of Aerodynamic Flows*. 1. Aufl. Berlin Heidelberg: Springer, 1999. ISBN: 978-3-540-66181-8.

[3] Charles R. Harris u. a. „Array programming with NumPy“. In: *Nature* 585.7825 (Sep. 2020), S. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[4] J.L. Hess und A.M.O. Smith. „Calculation of Potential Flow About Arbitrary Bodies“. In: *Progress in Aerospace Sciences* 8 (1966), S. 117–137.

[5] J. D. Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in Science & Engineering* 9.3 (2007), S. 90–95. DOI: 10.1109/MCSE.2007.55.

[6] Pauli Virtanen u. a. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“. In: *Nature Methods* 17 (2020), S. 261–272. DOI: 10.1038/s41592-019-0686-2.

Anhang A: Lednicer- und Selig-Format

Es folgt eine Gegenüberstellung des Selig-Formats (links) mit dem Lednicer-Format (rechts) am Beispiel des NACA M13-Profils.

```
NACA M13 AIRFOIL
1.00000000 0.00440000
0.95000000 0.01056000
0.90000000 0.01642000
0.80000000 0.02924000
0.70000000 0.04186000
0.60000000 0.05378000
0.50000000 0.06360000
0.40000000 0.06942000
0.30000000 0.07014000
0.20000000 0.06276000
0.15000000 0.05542000
0.10000000 0.04468000
0.07500000 0.03791000
0.05000000 0.02964000
0.02500000 0.01937000
0.01250000 0.01278000
0.00000000 0.00000000
0.00000000 0.00000000
0.01250000 -0.00812000
0.02500000 -0.00813000
0.05000000 -0.00791000
0.07500000 -0.00439000
0.10000000 -0.00222000
0.15000000 0.00192000
0.20000000 0.00506000
0.30000000 0.00804000
0.40000000 0.00772000
0.50000000 0.00550000
0.60000000 0.00228000
0.70000000 -0.00064000
0.80000000 -0.00226000
0.90000000 -0.00228000
0.95000000 -0.00144000
1.00000000 0.00000000
```

```
NACA M13 AIRFOIL
17 17
0.000000 0.000000
0.012500 0.012780
0.025000 0.019370
0.050000 0.029640
0.075000 0.037910
0.100000 0.044680
0.150000 0.055420
0.200000 0.062760
0.300000 0.070140
0.400000 0.069420
0.500000 0.063600
0.600000 0.053780
0.700000 0.041860
0.800000 0.029240
0.900000 0.016420
0.950000 0.010560
1.000000 0.004400

0.000000 0.000000
0.012500 -0.008120
0.025000 -0.008130
0.050000 -0.007910
0.075000 -0.004390
0.100000 -0.002220
0.150000 0.001920
0.200000 0.005060
0.300000 0.008040
0.400000 0.007720
0.500000 0.005500
0.600000 0.002280
0.700000 -0.000640
0.800000 -0.002260
0.900000 -0.002280
0.950000 -0.001440
1.000000 0.000000
```

Anhang B: Beispielausgabe der Methode .write_panels()

Hier ist die Ausgabe der ersten 30 Zeilen der AirfoilProfile-Methode .write_panels() gezeigt, am Beispiel des NASA: HSNLF(1)-0213 Profils gezeigt.

```
X_i,Y_i,theta_i,l_i
0.9950043,-0.0009728,0.06055800732379477,0.010009748628212382
0.9850132,-0.0016043,0.06568588453475652,0.010012392145736168
0.97752025,-0.002109200000000003,0.07047228200753602,0.005007529407801845
0.9625369500000001,-0.00330425,0.08141266954462605,0.025054484997700546
0.937568,-0.00552425,0.09593394112296426,0.02508172911124744
0.92258845,-0.0069828,0.10270780128934252,0.005019149972854041
0.910107,-0.0083076,0.10650475541673622,0.020084001420035736
0.8876414,-0.010771800000000002,0.11144687148830609,0.025116819156891672
0.8626817,-0.01365475,0.11853981264059243,0.025134786388787998
0.8377253,-0.01677495,0.13021388185551733,0.025167464110831598
0.82275295,-0.018755849999999998,0.13816384340639531,0.00503831234641125
0.81027805,-0.02055885,0.1448782150472314,0.020170819682402617
0.79032025,-0.0235841,0.15598759087809566,0.02020137326445906
0.77784795,-0.02556705,0.16438466111254837,0.005056667726873056
0.76288615,-0.0283095,0.184653165616612,0.02536632781405301
0.73795785,-0.03345505,0.22232106375561403,0.025550337335150823
0.7130314,-0.0387352,0.19509280338564264,0.025413502189190737
0.6905846,-0.0425474,0.13433154712117715,0.02014367279519793
0.6781074,-0.04417745,0.11216297598033555,0.005023767722536545
0.6631284,-0.0456803,0.09755918696978584,0.025085082682741986
0.63815875,-0.0478474,0.0755680809368733,0.02504497608084302
0.61318205,-0.0495071,0.05712775250964519,0.025020717455141107
0.58819985,-0.05077395,0.04420261223196785,0.025008928130969513
0.56321345,-0.05173975,0.03306342585001139,0.025001964705598602
0.5382233000000001,-0.05243635,0.02267137098916197,0.024998424208137613
0.5132296000000001,-0.0528728,0.012249641308583322,0.024997275443535843
0.48823320000000003,-0.0531112,0.006824603815373697,0.0249979821409649
0.4632347,-0.05320265000000004,0.0004920078324255504,0.02499960302584824
0.43823365000000003,-0.05310760000000005,6.275090293523316,0.025003319219855574
```

Anhang C: Codeausschnitte

Hier sind ausgewählte Codeausschnitte gezeigt, welche zur Lösung der Problemstellungen geschrieben wurden.

Die Klasse Panel

Die Klasse Panel modelliert die Panels $\mathcal{C}_i$ eines gegebenen Profils. Gespeichert werden neben den charakteristischen Parametern X_i, Y_i, θ_i, l_i auch der Normalwinkel des Panels δ_i, welcher aus dem Profil herauszeigt. Ebenso wird die Panelposition als Ober- oder Unterseite bestimmt (für einige besondere Profile ist auch ein Wert "verticalfür komplett senkrechte Panele möglich). Ebenso gespeichert wird die Quellbelegung q_i, die Tangentialgeschwindigkeit $v_i^{(t)}$ unter gegebenen Anströmwinkel α und -geschwindigkeit V_∞, und der resultierende Druckbeiwert c_{p_i}.

class Panel:

```
def __init__(self, xa, ya, xb, yb):
    self.xa, self.ya = xa, ya  # panel starting-point
    self.xb, self.yb = xb, yb  # panel ending-point

    self.xm = (xa + xb) / 2 # center of panel on x axis
    self.ym = (ya + yb) / 2 # center of panel on y axis
# panel length
    self.length = np.sqrt((xb - xa) ** 2 + (yb - ya) ** 2)

    self.theta = atan2(yb - ya, xb - xa) # angle of panel
    # if angle is negative, add 2pi to only have positive angles for
        plotting
    if self.theta < 0:
        self.theta += 2 * np.pi

    # normal angle of panel
    self.delta = self.theta - np.pi / 2
    if self.delta < 0:
        self.delta += 2 * np.pi
    if self.delta > 2 * np.pi:
        self.delta -= 2 * np.pi

    # panel location (used for plotting)
    if np.pi / 2 < self.theta < 3 * np.pi / 2:
        self.loc = 'upper'  # upper surface
```

```
        elif self.theta == np.pi / 2 or self.theta == 3 * np.pi / 2:
            self.loc = 'vertical'
        else:
            self.loc = 'lower'   # lower surface

        self.q = None # source strength
        self.vt = None # tangential velocity
        self.cp = None # pressure coefficient
```

Die Klasse AirfoilProfile

Die Klasse AirfoilProfile modelliert ein gegebenes Profil. Sie speichert neben wählbaren Namen und der ihr zugewiesenen Panele auch die Profiltiefe t und sämtliche Systemparameter $\xi_{ij},\ \eta_{ij},\ I_{ij},\ J_{ij},\ A_{ij}^{(n)},\ A_{ij}^{(t)},\ M_{ij}$.
Durch einen Aufruf der Methode .solve(V, a) mit gegebenen α und V_∞ werden für alle zugeordneten Panele die Quellstärken, Tangentialgeschwindigkeiten und Druckbeiwerte berechnet (und in den jeweiligen Klassenvariablen abgespeichert). Dabei wird ebenfalls die Genauigkeit der Approximation $\sum q_i l_i$ berechnet. Die Methode kann mit dem Schlüsselwortargument vortex=True aufgerufen werden, wodurch ebenfalls der Auftriebsbeiwert c_a, sowie die Wirbelbewegung γ
berechnet werden. Die Methode .write_panels() erzeugt eine .csv-Datei, welche für jedes Panel die Werte $X_i, Y_i \theta_i, l_i$ in eine Zeile schreibt (siehe Anhang A: Lednicer- und Selig-Format).
Die Methode .compute_free_vt(x, y, V, a) ermöglicht die Berechnung der Tangentialgeschwindigkeiten an jedem Punkt (x_i, y_i) des Profils. Diese werden als Tupel von der Methode zurückgegeben.

```
class AirfoilProfile :
    def __init__(self, panels, name, vortex=True):
        self.panels = panels
        self.name = name
        self.len = len(self.panels)
        self.vortex = vortex

        self.U = sum([panel.length for panel in self.panels])
        self.xi = compute_xi(self.len, self.panels)
        self.eta = compute_eta(self.len, self.panels)
        self.I = compute_I(self.len, self.xi, self.eta, self.panels)
        self.J = compute_J(self.len, self.xi, self.eta, self.panels)
        self.An = compute_An(self.len, self.I, self.J, self.panels)
        self.At = compute_At(self.len, self.I, self.J, self.panels)
```

```
        self.M = system_matrix(self.An, self.At, self.vortex)

        self.x = [panel.xa for panel in self.panels]
        self.y = [panel.ya for panel in self.panels]
        self.t = abs(max(self.x) - min(self.x))
        self.coords = [(panel.xa, panel.ya, panel.xb, panel.yb) for panel in
            self.panels]
        self.lower = [panel for panel in self.panels if panel.loc == "lower"]
        self.upper = [panel for panel in self.panels if panel.loc == "upper"]

        self.ca = None
        self.accuracy = None
        self.gamma = None

    def solve(self, V=1, a=np.radians(4.0)):
        b = compute_inhomogenity(self.panels, self.vortex, V, a)
        qs = np.linalg.solve(self.M, b)

        for i, panel in enumerate(self.panels):
            panel.q = qs[i]
        if self.vortex:
            self.gamma = qs[-1]

        if self.vortex:
            vt = []
            for i in range(self.len):
                vt.append(
                    sum([self.At[i][j] * self.panels[j].q for j in range(self.
                        len)]) - self.gamma * sum(
                        [self.An[i][j] for j in range(self.len)]) + V * np.cos
                            (
                        a - self.panels[i].theta))
        else:
            vt = []
            for i in range(self.len):
                vt.append(
                    sum([self.At[i][j] * self.panels[j].q for j in range(self.
                        len)]) + V * np.cos(
                        a - self.panels[i].theta))
        for i, panel in enumerate(self.panels):
```

```
            panel.vt = vt[i]

        for panel in self.panels:
            panel.cp = 1 - (panel.vt / V) ** 2

        self.ca = -2 / (V * self.t) * sum([panel.vt * panel.length for panel
            in self.panels])
        self.accuracy = sum([panel.q * panel.length for panel in self.panels])

    def write_panels(self):
        header = ["X_i", "Y_i", "theta_i", "l_i"]
        with open("data/vals/" + self.name + "_vals.csv", "w+", encoding='
            UTF8',newline="") as file:
            writer = csv.writer(file)
            writer.writerow(header)
            #file.write(f"Circumference: {self.U} \n")
            #file.write(f"X_i, Y_i, theta_i, l_i\n")
            for panel in self.panels:
                #file.write(f"{panel.xm}, {panel.ym}, {panel.theta}, {panel.
                    length}\n")
                writer.writerow([panel.xm, panel.ym, panel.theta, panel.length
                    ])

    def compute_free_vt(self, x, y, V=1, a=np.radians(4.0)):
        eta = compute_eta_free(len(x), self.len, self.panels, x, y)
        xi = compute_xi_free(len(x), self.len, self.panels, x, y)
        I = compute_I_free(len(x), self.len, xi, eta, self.panels)
        J = compute_J_free(len(x), self.len, xi, eta, self.panels)
        An = compute_An_free(len(x), self.len, I, J, self.panels)
        At = compute_At_free(len(x), self.len, I, J, self.panels)

        vtx = np.empty(len(x))
        vty = np.empty(len(x))
        for i in range(len(x)):
            vtx[i] = sum([At[i][j] * self.panels[j].q for j in range(self.len)
                ]) - self.gamma * sum(
                [An[i][j] for j in range(self.len)]) + V * np.cos(a)

            vty[i] = sum([An[i][j] * self.panels[j].q for j in range(self.len)
                ]) + self.gamma * sum(
```

```
                [At[i][j] for j in range(self.len)]) + V * np.sin(a)

        return vtx, vty
```

make_cylinder(r,n)

Diese Funktion wurde verwendet, um die x- und y-Koordinaten eines Kreiszylinder mit Radius r und n gleich langen Panels zu generieren.

```
def cylinder(r=1, n=8):
    # generate a cylinder with n equidistant panels
    a = np.linspace(0, 360, num=n+1, endpoint=True) / 180 * np.pi

    x = r * np.cos(a)
    y = r * np.sin(a)

    return x, y
```