

任意形状物体绕流的数值计算方法《非主要内容》

阅前说明

本文档为附加的涡面法理论推导与数值求解详细思路，旨在完整展示本程序的底层数学逻辑与代码构建过程。

若需查阅主要报告，敬请参阅同文件夹下的主报告文档：《空气动力学基础期中大作业报告《主要内容》》

第一部分：基本信息

题目：NACA四位翼型的数值计算

作者信息：张启民 2300098603

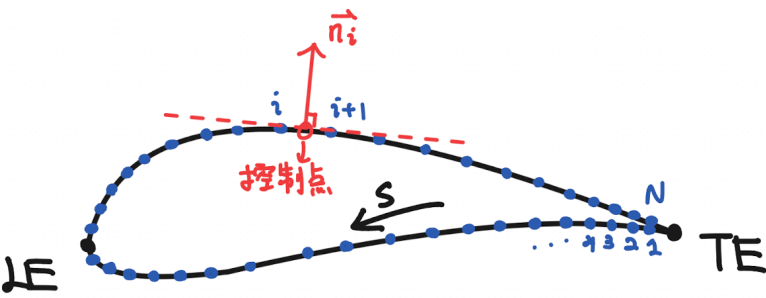
基本思路

在涡面法的数值求解中，将翼型表面离散化后会产生 n 个未知的节点环量 γ_i （其中 $i = 1, 2, \dots, n$ ）。通过在翼型表面的控制点上应用无穿透边界条件（即法向速度为零），可以建立 $n - 1$ 个线性方程；同时，引入库塔条件作为第 n 个补充约束。最终，这 n 个未知量与 n 个独立方程构成了一个封闭的线性代数方程组，可将其转化为矩阵形式 $[A][x] = [B]$ 进行求解。

目录

- 第一部分：基本信息
- 第二部分：详细思路
- 第三部分：网格划分
- 第四部分：矩阵 $[P]_{ij}$ 的计算
- 第五部分：矩阵 $[A]$ 的计算
- 第六部分：向量 $[B]$ 的计算
- 第七部分：气动系数的推导与计算

第二部分：详细思路



1. 几何离散化

- 把翼型上下表面分割成有限个微元面。

- 由 N 个节点分段。
- 形成 $N - 1$ 个微元面。
- 设微元面的中点为控制点，则有 $N - 1$ 个控制点。

2. 线涡强度定义

- 记节点上的线涡强度为 γ_i 。
- 假设每个微元面上的线涡强度是线性分布的。

3. 约束方程与边界条件

设控制点 i 上的速度为 $\vec{U}_i$ ，法向量为 $\vec{n}_i$ 则必须满足以下两个核心条件：

- **(1) 无穿透边界条件：** 速度法向分量为 0，即：

$$\vec{U}_i \cdot \vec{n}_i = 0 \quad (7)$$

此条件可提供 $N - 1$ 个方程。

- **(2) Kutta 条件：** 为保证流体平滑地从尾缘离开，尾缘上、下表面的切向速度大小必须相等。由于环量积分路径在尾缘上、下表面的方向相反，为了使真实物理速度一致，尾缘节点处的净涡强必须为 0。离散后表示为：

$$\gamma_1 + \gamma_N = 0 \quad (8)$$

此条件可提供 1 个方程。

推论： 结合以上两个条件，我们共有 N 个方程，可以求解 N 个未知数（即 N 个节点的线涡强度 γ ）。

4. 扰动速度与矩阵推导

控制点 i 上的总速度方程为：

$$\vec{U}_i = \vec{U}_\infty + \sum_{j=1}^{N-1} \vec{U}_{ji} \quad (9)$$

(注： $\vec{U}_{ji}$ 为由微元 j 在控制点 i 上引起的扰动速度)

由于微元 j 是由节点 j 和节点 $j + 1$ 构成的，其上的速度与 γ_j 和 γ_{j+1} 相关。因此，在控制点 i 上引起的扰动速度可以写为矩阵形式：

$$\vec{U}_{ji} = [P]_{ji} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix}$$

将方程 (3) 代入边界条件方程 (1) 中，可得：

$$\vec{U}_i \cdot \vec{n}_i = \vec{U}_\infty \cdot \vec{n}_i + \sum_{j=1}^{N-1} \vec{U}_{ji} \cdot \vec{n}_i = 0$$

移项整理后得到：

$$\sum_{j=1}^{N-1} \vec{U}_{ji} \cdot \vec{n}_i = -\vec{U}_\infty \cdot \vec{n}_i$$

带入 $\vec{U}_{ji}$ 和 γ 的表达式：

$$\sum_{j=1}^{N-1} [P]_{ji} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix} \cdot \vec{n}_i = -\vec{U}_\infty \cdot \vec{n}_i$$

5. 整合与求解

考虑 $N - 1$ 个控制点，就会得到 $N - 1$ 个方程，并将方程写成矩阵形式：

$$[A_1]_{(N-1) \times N} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_N \end{bmatrix} = U_\infty [B_1]_{(N-1) \times 1} \quad (10)$$

接下来，考虑 Kutta 条件即方程 (2)：

$$\gamma_1 + \gamma_N = 0$$

将其写为矩阵的一行：

$$\begin{bmatrix} 1 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_N \end{bmatrix} = 0 \quad (11)$$

最后，将方程 (4) 和 (5) 整合在一起，构建出最终的 $N \times N$ 线性方程组：

$$\begin{bmatrix} & [A_1] \\ 1 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_N \end{bmatrix} = U_\infty \begin{bmatrix} B_1 \\ 0 \end{bmatrix}$$

这即为标准的线性代数求解形式：

$$A_{N \times N} \gamma_{nondim} = B_{N \times 1} \quad (12)$$

(注：这里 $\gamma_{nondim} = \frac{\gamma}{U_\infty}$ 是无量纲化的线涡强密度)

第三部分：划分网格（微元面）

在涡面法中，为了更好地捕捉前缘和尾缘处剧烈的流动变化，通常采用半余弦划分法对翼型表面进行节点离散。

(注：节点编号通常从翼型下表面的尾缘开始 ($i = 1$)，逆时针绕过前缘，最终结束于上表面的尾缘 ($i = N$)。)

根据节点总数 N 的奇偶性，划分方法分为以下两种情况：

1. 当 N 为奇数时（前缘处为节点）

此时，离散节点会精确落在翼型的前缘点上。

- 角度增量：

$$\Delta\theta = \frac{\pi}{N/2}$$

- 节点 x 坐标分布（对于 $i = 1, 2, \dots, N/2$ ）：

$$\frac{x_i}{c} = \frac{1}{2}(1 - \cos(i\Delta\theta))$$

2. 当 N 为偶数时（前缘处为控制点）

此时，前缘点被两个节点包围，前缘本身作为微元的控制点。

- 角度增量：

$$\Delta\theta = \frac{\pi}{N/2 - 1/2}$$

- 节点 x 坐标分布（对于 $i = 1, 2, \dots, N/2$ ）：

$$\frac{x_i}{c} = \frac{1}{2} \left[1 - \cos \left(i\Delta\theta - \frac{1}{2}\Delta\theta \right) \right]$$

（注：计算出 x_i 后，再代入翼型的厚度与中弧线几何方程，即可得到翼型上下表面的实际 (x, y) 坐标点，完成从尾缘出发、绕过前缘、再回到尾缘的完整离散。）

#计算出翼型上表面和下表面的坐标的函数

```
def geometry(x):
    camber = np.zeros(2)
    upper = np.zeros(2)
    lower = np.zeros(2)

    if x == 0.0:
        pass
    else:
        if x < p:
            yc = m*(2.0*(x/p) - (x/p)**2)
            dydx = 2.0*m/p*(1.0 - x/p)
        else:
            yc = m*(2.0*(1.0-x)/(1.0-p) - ((1.0-x)/(1.0-p))**2)
            dydx = -2.0*m/(1.0-p)*(1.0 - (1.0-x)/(1.0-p))

        camber[0] = x
        camber[1] = yc
        thick = t*(2.969*np.sqrt(x) - 1.260*x - 3.516*x**2 + 2.843*x**3 -
1.015*x**4)

        denom = np.sqrt(1.0 + dydx**2)
        upper[0] = x - thick / (2.0 * denom) * dydx
        upper[1] = yc + thick / (2.0 * denom)
        lower[0] = x + thick / (2.0 * denom) * dydx
        lower[1] = yc - thick / (2.0 * denom)

    return camber, upper, lower
```

```

#计算出法向量的函数
def normalize(v):
    mag = np.sqrt(v[0]**2 + v[1]**2)
    return v / mag

#网格划分函数
def meshing():
    np2 = int(npts/2)
    #偶数情况
    if npts % 2 == 0:
        dtheta = 2.0*pi/float(npts-1)
        for i in range(np2):
            x = 0.5*(1.0-np.cos((float(i+1)-0.5)*dtheta))
            camber, Points[np2+i,:], Points[np2-i-1,:] = geometry(x)
    #奇数情况
    else:
        dtheta = pi/float(np2)
        for i in range(np2):
            x = 0.5*(1.0-np.cos(float(i+1)*dtheta))
            camber, Points[np2+1+i,:], Points[np2-1-i,:] = geometry(x)
    #划分控制点及其法向量
    for i in range(npts-1):
        Control[i,0] = 0.5*(Points[i,0] + Points[i+1,0])
        Control[i,1] = 0.5*(Points[i,1] + Points[i+1,1])
        Normal[i,0] = - (Points[i+1,1] - Points[i,1])
        Normal[i,1] = (Points[i+1,0] - Points[i,0])
        Normal[i,:] = normalize(Normal[i,:])

```

3. 网格无关性验证与最优节点数

在确定了网格划分方法后，需通过**网格无关性验证求出最适合的节点数 n_{best} 。

- 随着节点数 n 的增加，计算得到的升力系数 c_l 会逐渐趋于稳定。
- 当 n 达到某一个临界值 n_{best} 后， c_l 的变化率极小。此时的 n_{best} 即为兼顾计算精度与计算效率的最佳网格数量。

以下是在特定参数，满足误差精度及计算效率的 $n_{best} = 250$

```

target_alpha = 5.0
n_list = np.arange(20, 500, 10)
tolerance = 1e-5 #容差

#=====结果=====

```

节点数 (N)	升力系数 (c_l)	变化量 (Delta c_l)
20	1.106196	N/A
30	1.115506	0.009310
40	1.118893	0.003387
50	1.120487	0.001594
60	1.121354	0.000866
70	1.121874	0.000520
80	1.122209	0.000336

90	1.122439	0.000230
100	1.122604	0.000164
110	1.122724	0.000120
120	1.122815	0.000091
130	1.122887	0.000071
140	1.122943	0.000057
150	1.122989	0.000046
160	1.123026	0.000037
170	1.123057	0.000031
180	1.123083	0.000026
190	1.123105	0.000022
200	1.123123	0.000019
210	1.123140	0.000016
220	1.123153	0.000014
230	1.123165	0.000012
240	1.123176	0.000011
250	1.123185	0.000009
#在n=250时, Delta c1已经达到1e-6级别了		
260	1.123194	0.000008
270	1.123201	0.000007
280	1.123208	0.000007
290	1.123214	0.000006
300	1.123219	0.000005
310	1.123224	0.000005
320	1.123228	0.000004
330	1.123232	0.000004
340	1.123236	0.000004
350	1.123239	0.000003
360	1.123243	0.000003
370	1.123245	0.000003
380	1.123248	0.000003
390	1.123250	0.000002
400	1.123253	0.000002
410	1.123255	0.000002
420	1.123257	0.000002
430	1.123258	0.000002
440	1.123260	0.000002
450	1.123262	0.000002
460	1.123263	0.000001
470	1.123264	0.000001
480	1.123266	0.000001
490	1.123267	0.000001
500	1.123268	0.000001

>>> 结论：根据容差 1e-05，推荐的最佳节点数为 n = 250

第四部分：矩阵 $[P]_{ji}$ 的计算

微元面 $(j, j + 1)$ 在控制点 i 处产生的诱导速度可表示为矩阵形式：

$$\vec{U}_{ji} = [P]_{ji} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix}$$

1. 坐标变换：全局坐标 $(X, Y) \rightarrow$ 局部坐标 (ξ, η)

为了方便积分，将微元面转换到局部坐标系下。变换过程包含平移与旋转：

- **平移：** 将坐标原点从 $(0, 0)$ 平移至微元面起点 (X_j, Y_j) ，即 $\begin{bmatrix} X - X_j \\ Y - Y_j \end{bmatrix}$
- **旋转：** 根据微元的几何关系， $\cos \theta = \frac{X_{j+1} - X_j}{l_j} = \frac{\Delta X}{l_j}$ ， $\sin \theta = \frac{Y_{j+1} - Y_j}{l_j} = \frac{\Delta Y}{l_j}$ 。旋转矩阵为：

$$\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \frac{1}{l_j} \begin{bmatrix} \Delta X & \Delta Y \\ -\Delta Y & \Delta X \end{bmatrix}$$

综合平移与旋转，坐标变换方程为：

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \frac{1}{l_j} \begin{bmatrix} \Delta X & \Delta Y \\ -\Delta Y & \Delta X \end{bmatrix} \begin{bmatrix} X - X_j \\ Y - Y_j \end{bmatrix}$$

2. 微元面 $(j, j+1)$ 在 (ξ, η) 点上产生的诱导速度

设线涡强度密度在微元面上是线性分布的，即：

$$\gamma(s) = \gamma_j + \frac{\gamma_{j+1} - \gamma_j}{l_j} s$$

由定义，点涡在 (ξ, η) 上产生的诱导速度为 $dU = \frac{\gamma(s)ds}{2\pi r}$ ，其中 $r = \sqrt{\eta^2 + (\xi - s)^2}$

则可以分解为：

$$\begin{cases} dU_\xi = dU \cdot \frac{\eta}{r} = \frac{\eta\gamma(s)}{2\pi((\xi-s)^2 + \eta^2)} ds \\ dU_\eta = -dU \cdot \frac{\xi-s}{r} = \frac{-(\xi-s)\gamma(s)}{2\pi((\xi-s)^2 + \eta^2)} ds \end{cases}$$

求解 U_ξ 的积分过程：

$$U_\xi = \int_0^{l_j} \frac{\eta \left(\gamma_j + \frac{\gamma_{j+1} - \gamma_j}{l_j} s \right)}{2\pi((\xi - s)^2 + \eta^2)} ds \quad ; \text{令 } \Delta\gamma = \gamma_{j+1} - \gamma_j$$

$$2\pi U_\xi = \gamma_j \int_0^{l_j} \frac{\eta}{(\xi - s)^2 + \eta^2} ds + \frac{\Delta\gamma}{l_j} \int_0^{l_j} \frac{s\eta}{(\xi - s)^2 + \eta^2} ds$$

第一项积分：令 $\Phi = \int_0^{l_j} \frac{\eta}{(\xi - s)^2 + \eta^2} ds$

第二项积分：令 $s = \xi - (\xi - s)$ ，即：

$$\begin{aligned} \int_0^{l_j} \frac{s\eta}{(\xi - s)^2 + \eta^2} ds &= \xi \int_0^{l_j} \frac{\eta}{(\xi - s)^2 + \eta^2} ds - \eta \int_0^{l_j} \frac{\xi - s}{(\xi - s)^2 + \eta^2} ds \\ &= \xi\Phi - \eta\Psi \quad ; \text{其中令 } \Psi = \int_0^{l_j} \frac{\xi - s}{(\xi - s)^2 + \eta^2} ds \end{aligned}$$

代回原式可得：

$$2\pi U_\xi = \gamma_j \Phi + \frac{\Delta\gamma}{l_j} (\xi\Phi - \eta\Psi)$$

求解 U_η 的积分过程：

$$\begin{aligned} \text{同理 } U_\eta &= \int_0^{l_j} \frac{-(\xi-s) \left(\gamma_j + \frac{\Delta\gamma}{l_j} s \right)}{2\pi[(\xi-s)^2 + \eta^2]} ds \\ 2\pi U_\eta &= -\gamma_j \underbrace{\int_0^{l_j} \frac{\xi-s}{(\xi-s)^2 + \eta^2} ds}_{\Psi} - \frac{\Delta\gamma}{l_j} \int_0^{l_j} \frac{s(\xi-s)}{(\xi-s)^2 + \eta^2} ds \end{aligned}$$

对第二项积分进行代换整理（代入 $s = \xi - (\xi - s)$ ）：

$$\begin{aligned} \int_0^{l_j} \frac{s(\xi-s)}{(\xi-s)^2 + \eta^2} ds &= \xi \underbrace{\int_0^{l_j} \frac{\xi-s}{(\xi-s)^2 + \eta^2} ds}_{\Psi} - \int_0^{l_j} \frac{(\xi-s)^2}{(\xi-s)^2 + \eta^2} ds \\ &\hookrightarrow \text{对后面部分分子加减 } \eta^2 \text{ (即 } +\eta^2 - \eta^2) \\ &= \xi\Psi + \int_0^{l_j} 1 ds - \eta \int_0^{l_j} \frac{\eta}{(\xi-s)^2 + \eta^2} ds \\ &= \xi\Psi + l_j - \eta\Phi \end{aligned}$$

代回原式可得：

$$2\pi U_\eta = -\gamma_j \Psi - \frac{\Delta\gamma}{l_j} (\xi\Psi - l_j + \eta\Phi)$$

计算 Φ 与 Ψ 的解析解：

$$\begin{aligned} \Phi &= \int_0^{l_j} \frac{\eta}{(\xi-s)^2 + \eta^2} ds = \left[\arctan \left(\frac{s-\xi}{\eta} \right) \right]_0^{l_j} = \arctan \left(\frac{l_j-\xi}{\eta} \right) + \arctan \left(\frac{\xi}{\eta} \right) \\ &= \arctan \left(\frac{\eta l_j}{\eta^2 - \xi l_j + \xi^2} \right) = \text{atan2}(\eta l_j, \eta^2 + \xi^2 - \xi l_j) \\ \Psi &= \int_0^{l_j} \frac{\xi-s}{(\xi-s)^2 + \eta^2} ds = \left[-\frac{1}{2} \ln((\xi-s)^2 + \eta^2) \right]_0^{l_j} \\ &= \frac{1}{2} \ln \left[\frac{\xi^2 + \eta^2}{(\xi-l_j)^2 + \eta^2} \right] \end{aligned}$$

4. 转换回全局坐标系并组装 $[P]_{ji}$ 矩阵

稍微整理一下展开项：

$$\begin{cases} 2\pi l_j U_\xi = [(l_j - \xi)\Phi + \eta\Psi]\gamma_j + [\xi\Phi - \eta\Psi]\gamma_{j+1} \\ 2\pi l_j U_\eta = [\eta\Phi - (l_j - \xi)\Psi - l_j]\gamma_j + [-\eta\Phi - \xi\Psi + l_j]\gamma_{j+1} \end{cases}$$

写成矩阵形式：

$$\begin{bmatrix} U_\xi \\ U_\eta \end{bmatrix} = \frac{1}{2\pi l_j} \begin{bmatrix} (l_j - \xi)\Phi + \eta\Psi & \xi\Phi - \eta\Psi \\ \eta\Phi - (l_j - \xi)\Psi - l_j & -\eta\Phi - \xi\Psi + l_j \end{bmatrix} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix}$$

转换至全局坐标系 (X, Y) ：

由微元面 $(j, j+1)$ 在点 (X, Y) 上所产生的诱导速度分布为 $\vec{U}_{ji}$ ：

$$\vec{U}_{ji} = \begin{bmatrix} U_X \\ U_Y \end{bmatrix} = \frac{1}{l_j} \begin{bmatrix} \Delta X & -\Delta Y \\ \Delta Y & \Delta X \end{bmatrix} \begin{bmatrix} U_\xi \\ U_\eta \end{bmatrix}$$

代入局部坐标系矩阵结果：

$$= \frac{1}{2\pi l_j^2} \begin{bmatrix} \Delta X & -\Delta Y \\ \Delta Y & \Delta X \end{bmatrix} \begin{bmatrix} (l_j - \xi)\Phi + \eta\Psi & \xi\Phi - \eta\Psi \\ \eta\Phi - (l_j - \xi)\Psi - l_j & -\eta\Phi - \xi\Psi + l_j \end{bmatrix} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix}$$

最终可以简写为：

$$= [P]_{ji} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix}$$

(注：矩阵 $[P]_{ji}$ 只与几何有关)

那么可以得到：

$$[P]_{ji} = \frac{1}{2\pi l_j^2} \begin{bmatrix} \Delta X & -\Delta Y \\ \Delta Y & \Delta X \end{bmatrix} \begin{bmatrix} (l_j - \xi)\Phi + \eta\Psi & \xi\Phi - \eta\Psi \\ \eta\Phi - (l_j - \xi)\Psi - l_j & -\eta\Phi - \xi\Psi + l_j \end{bmatrix}$$

```
def P_mat(point1, point2, poi):
    A = np.zeros((2,2))
    B = np.zeros((2,2))

    li = np.sqrt((point2[0]-point1[0])**2 + (point2[1]-point1[1])**2)
    dx = point2[0]-point1[0]; dy = point2[1]-point1[1]

    A[0,0] = dx; A[0,1] = dy
    A[1,0] = -dy; A[1,1] = dx
    vec1 = poi - point1
    vec2 = 1.0/li * np.matmul(A, vec1)
    xi = vec2[0]; eta = vec2[1]

    phi = np.arctan2(eta*li, eta**2 + xi**2 - xi*li)
    psi = 0.5*np.log((xi**2 + eta**2)/((xi-li)**2 + eta**2))

    A[0,0] = dx; A[0,1] = -dy
    A[1,0] = dy; A[1,1] = dx

    B[0,0] = (li-xi)*phi + eta*psi; B[0,1] = xi*phi - eta*psi
    B[1,0] = eta*phi - (li-xi)*psi - li; B[1,1] = -eta*phi - xi*psi + li

    P = 1.0/(2.0*pi*li**2) * np.matmul(A,B)
    return P
```

第五部分：矩阵 $[A]$ 的计算

由边界条件可知，所有微元面在控制点 i 处产生的诱导速度法向分量之和等于总系数矩阵 $[A_1]$ 与线涡强密度向量 $[\gamma]$ 的乘积：

$$\sum_{j=1}^{n-1} \vec{U}_{ji} \cdot \vec{n}_i = \sum_{j=1}^{n-1} \left([P]_{ji} \begin{bmatrix} \gamma_j \\ \gamma_{j+1} \end{bmatrix} \right) \cdot \vec{n}_i = [A_1][\gamma]$$

其中，将矩阵 $[P]_{ji}$ 的四个元素展开表示为：

$$[P]_{ji} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{ji}$$

则全局坐标系下的诱导速度分量可以写为：

$$\begin{cases} U_X = P_{00}\gamma_j + P_{01}\gamma_{j+1} \\ U_Y = P_{10}\gamma_j + P_{11}\gamma_{j+1} \end{cases}$$

控制点 i 处的单位法向量 $\vec{n}_i$ 定义为：

$$\vec{n}_i = \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \frac{1}{l_i} \begin{bmatrix} -(Y_2 - Y_1) \\ X_2 - X_1 \end{bmatrix}$$

将诱导速度与法向量进行点乘：

$$\vec{U}_{ji} \cdot \vec{n}_i = U_X n_x + U_Y n_y$$

代入前面的速度展开式：

$$= (P_{00}\gamma_j + P_{01}\gamma_{j+1})(n_x) + (P_{10}\gamma_j + P_{11}\gamma_{j+1})(n_y)$$

提取公因式 γ_j 和 γ_{j+1} 进行合并整理：

$$= (P_{00}n_x + P_{10}n_y)\gamma_j + (P_{01}n_x + P_{11}n_y)\gamma_{j+1}$$

由此可以得出组装系数矩阵 $[A_1]$ 的核心累加逻辑：

$$\Rightarrow \sum_{j=1}^{n-1} \vec{U}_{ji} \cdot \vec{n}_i = [A_1][\gamma]$$

在编程循环中，矩阵元素的更新方式为：

$$\text{其中 } \begin{cases} A[i, j] = A[i, j] + (P_{00}n_x + P_{10}n_y) \\ A[i, j+1] = A[i, j+1] + (P_{01}n_x + P_{11}n_y) \end{cases}$$

该循环的遍历范围为：

- 外层循环：从 $i = 1 \rightarrow i = n - 1$ （代表共有 $n - 1$ 个控制点）
- 内层循环：从 $j = 1 \rightarrow j = n - 1$ （代表共有 $n - 1$ 个微元面）

补充 Kutta 条件

矩阵 A 的最后一行需要用来满足 Kutta 条件：

$$\gamma_1 + \gamma_n = 0$$

对应到代码的矩阵元素赋值中，即：

$$A[n, 1] = 1, \quad A[n, n] = 1$$

最终，将无穿透边界条件矩阵 $[A_1]$ 与 Kutta 条件组合，形成完整的系统求解矩阵 $[A]$ ：

$$[A] = \begin{bmatrix} & & [A_1] & & \\ 1 & 0 & \dots & 0 & 1 \end{bmatrix}$$

```
def setup():
```

```

global Amat
meshing()
for i in range(npts-1):
    x1, y1 = Points[i,0], Points[i,1]
    x2, y2 = Points[i+1,0], Points[i+1,1]
    li = np.sqrt((x2-x1)**2 + (y2-y1)**2)
    for j in range(npts-1):
        P = P_mat(Points[j,:], Points[j+1,:], Control[i,:])
        Amat[i,j] = Amat[i,j] + (x2-x1)/li*P[1,0] - (y2-y1)/li*P[0,0]
        Amat[i,j+1] = Amat[i,j+1] + (x2-x1)/li*P[1,1] - (y2-y1)/li*P[0,1]

Amat[npts-1,0] = 1.0
Amat[npts-1,npts-1] = 1.0

```

第六部分：向量 B 的计算

由前面整理的无穿透边界条件方程可知，方程的右端项为：

$$-\vec{U}_{\infty} \cdot \vec{n}_i = U_{\infty} [B_1]$$

其中，给定攻角为 α ，自由流速度向量 $\vec{U}_{\infty}$ 可以分解为：

$$\vec{U}_{\infty} = U_{\infty} \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix}$$

将自由流速度与控制点 i 的单位法向量 $\vec{n}_i$ 进行点乘：

$$\begin{aligned}
 -\vec{U}_{\infty} \cdot \vec{n}_i &= -U_{\infty} \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \end{bmatrix} \\
 &= -U_{\infty} (n_x \cos \alpha + n_y \sin \alpha)
 \end{aligned}$$

由于我们在方程右侧提取了自由流速度大小 U_{∞} 作为公因数，因此向量 $[B_1]$ 中第 i 个元素（即对应第 i 个控制点的值）的计算公式为：

$$B[i] = -n_x \cos \alpha - n_y \sin \alpha \quad ; \quad \text{其中 } \vec{n}_i = \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \frac{1}{l_i} \begin{bmatrix} -(Y_2 - Y_1) \\ X_2 - X_1 \end{bmatrix}$$

最后，与矩阵 A 的组装逻辑一致，我们需要在向量的最末尾加上一行来满足 Kutta 条件（其方程右侧的值为 0）。因此，最终的全局向量 $[B]$ 形式为：

$$[B] = \begin{bmatrix} [B_1] \\ 0 \end{bmatrix} \leftarrow \text{Kutta 条件}$$

```

def solve():
    global nondim_Gamma
    B = np.zeros(npts)
    for i in range(npts-1):
        x1, y1 = Points[i,0], Points[i,1]
        x2, y2 = Points[i+1,0], Points[i+1,1]
        li = np.sqrt((x2-x1)**2 + (y2-y1)**2)
        B[i] = ((y2-y1) * np.cos(alpha) - (x2-x1) * np.sin(alpha)) / li
    B[npts-1] = 0.0
    nondim_Gamma = np.linalg.solve(Amat, B)

```

第七部分：气动系数的推导与计算

在通过求解矩阵方程 $[A][\gamma_{nondim}] = [B]$ 得到所有节点上的无量纲线涡强度 γ_{nondim} 之后，我们就可以进一步求解翼型表面的压力分布与总升力。

1. 表面压力系数 (C_p) 的推导

压力系数 C_p 的推导基于不可压缩流体中的伯努利方程 (Bernoulli's Equation)。

对于流场中远前方未受扰动的自由流点 (状态为 p_∞, U_∞) 与翼型表面上的任意控制点 i (状态为 p_i, U_i)，沿着流线应用伯努利方程：

$$p_\infty + \frac{1}{2}\rho U_\infty^2 = p_i + \frac{1}{2}\rho U_i^2$$

将等式重新排列，求解压差 ($p_i - p_\infty$)：

$$p_i - p_\infty = \frac{1}{2}\rho U_\infty^2 - \frac{1}{2}\rho U_i^2 = \frac{1}{2}\rho(U_\infty^2 - U_i^2)$$

根据压力系数的无量纲化定义：

$$C_{p,i} = \frac{p_i - p_\infty}{\frac{1}{2}\rho U_\infty^2}$$

将压差代入定义式，即可得到 C_p 与速度的关系：

$$C_{p,i} = \frac{\frac{1}{2}\rho(U_\infty^2 - U_i^2)}{\frac{1}{2}\rho U_\infty^2} = 1 - \left(\frac{U_i}{U_\infty}\right)^2$$

由于在控制点 i 处满足无穿透边界条件 (法向速度为 0)，这里的 U_i 实际上就是该点处的切向诱导速度 (Tangential Velocity)。在代码实现中，由于速度通常已经用 U_∞ 进行了无量纲化处理 $U_{i,nondim} = \frac{U_i}{U_\infty}$ ，公式可以直接简化为：

$$C_{p,i} = 1 - U_{i,nondim}^2$$

```
def nondim_velocity(point):
    V = np.zeros(2)
    for i in range(npts-1):
        P = P_mat(Points[i,:], Points[i+1,:], point)
        V = V + np.matmul(P, nondim_Gamma[i:i+2])

    V[0] = V[0] + np.cos(alpha)
    V[1] = V[1] + np.sin(alpha)

    Cp = 1.0 - (V[0]**2 + V[1]**2)
    return V, Cp

Cp = np.zeros(npts-1)
offset = 0.001
for i in range(npts-1):
    V, Cp[i] = nondim_velocity(Control[i,:] + offset * Normal[i,:])
```

为什么在微元面的法向量上要有微小的offset呢？

现看看代码中的 `P_mat` 函数里计算角度 Φ 的代码：

```
phi = np.arctan2(eta*li, eta**2 + xi**2 - xi*li)
```

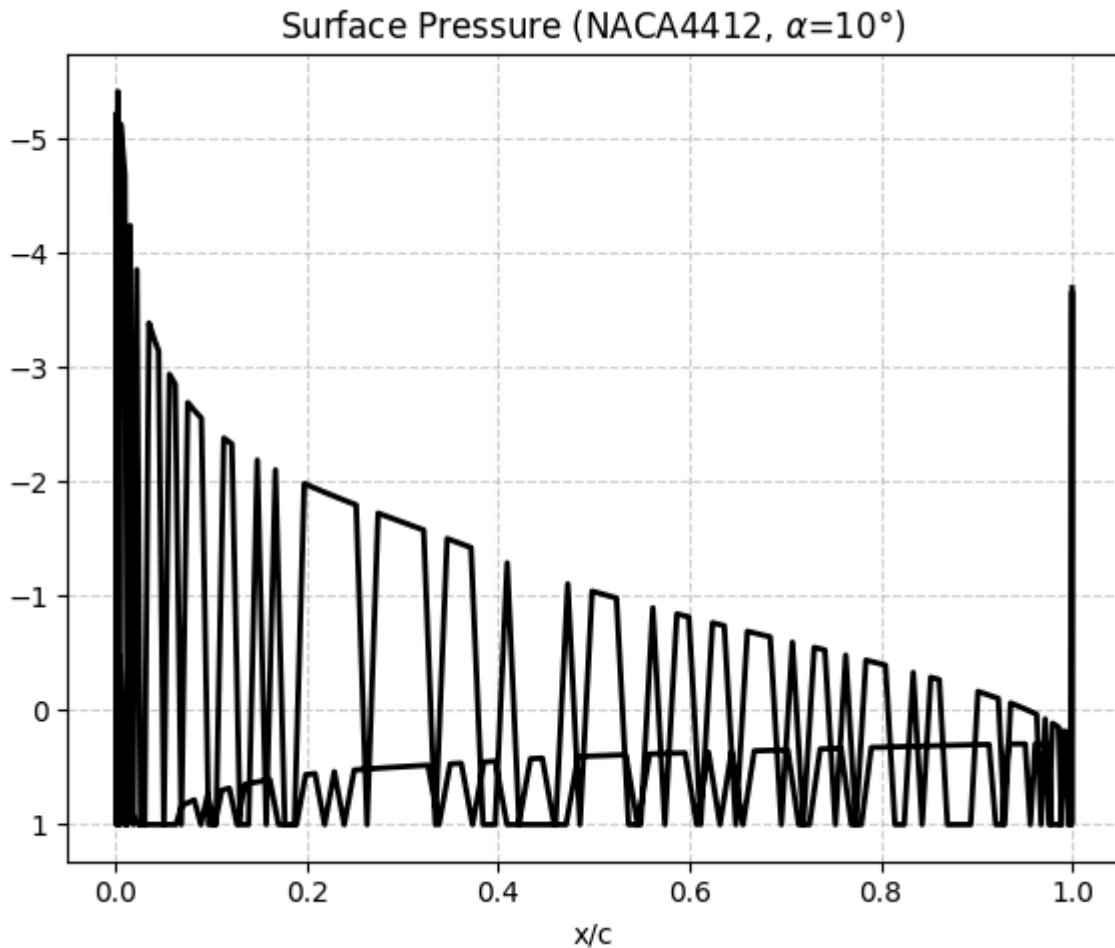
当计算面元自己对自己控制点的影响时，代入 $\xi = l_i/2$ 和 $\eta = 0$ ：

- 分子： $\eta \cdot l_i = 0$
- 分母： $0 + (l_i/2)^2 - (l_i/2) \cdot l_i = -\frac{1}{4}l_i^2$ （这是一个负数）

这个时候，计算机需要求解 `arctan2(0, 负数)`。

- 如果 η 是一个极其微小的正数 ($+10^{-16}$)，`arctan2` 的结果是 $+\pi$ 。
- 如果 η 是一个极其微小的负数 (-10^{-16})，`arctan2` 的结果是 $-\pi$ 。

在计算机有限的浮点数精度下，当直接把点放在表面上（理论上 $\eta = 0$ ），由于浮点误差，计算出来的 η 可能是 $+10^{-16}$ 也可能是 -10^{-16} 。这就导致计算出来的角度 Φ 在 $+\pi$ 和 $-\pi$ 之间疯狂横跳，进而导致算出来的翼型表面压强系数出现剧烈的随机误差。下图给出 `offset = 0`，计算机计算出的结果。



2. 总升力系数 (c_l) 的推导

升力系数的推导基于库塔-儒可夫斯基定理，该定理指出了二维无粘流中，升力 L 与环量 Γ 的直接关系：

$$L = \rho U_\infty \Gamma_{\text{total}}$$

总环量 Γ_{total} 等于翼型表面所有微元面 (Panel) 上环量 Γ_i 的积分之和。对于第 i 个微元面，由于线涡强度密度 γ 是线性分布的，其上的局部环量为：

$$\Gamma_i = \int_0^{l_i} \gamma(s) ds = \frac{\gamma_i + \gamma_{i+1}}{2} l_i$$

(即微元面两端线涡强密度的平均值乘以微元长度 l_i)

将所有微元的环量相加，得到总环量：

$$\frac{\Gamma_{\text{total}}}{U_{\infty}} = \sum_{i=1}^{n-1} \frac{\Gamma_i}{U_{\infty}} = \sum_{i=1}^{n-1} \frac{\gamma_{i,\text{nondim}} + \gamma_{i+1,\text{nondim}}}{2} l_i$$

接下来代入升力系数的标准定义式：

$$c_l = \frac{L}{\frac{1}{2} \rho U_{\infty}^2 c}$$

(注：式中 c 为翼型的弦长)

将 Kutta-Joukowski 定理 $L = \rho U_{\infty} \Gamma_{\text{total}}$ 代入上式，并且令 $x_{\text{nondim}} = \frac{x}{c}$ ，则在 $x_{\text{nondim}} \in [0, 1]$ 上的升力系数为：

$$\begin{aligned} c_l &= \frac{\rho U_{\infty} \Gamma_{\text{total}}}{\frac{1}{2} \rho U_{\infty}^2 c} = \frac{2 \Gamma_{\text{total}}}{U_{\infty} c} \\ &= 2 \sum_{i=1}^{n-1} \left(\frac{\gamma_{i,\text{nondim}} + \gamma_{i+1,\text{nondim}}}{2} l_i \right) = \sum_{i=1}^{n-1} l_i (\gamma_{i,\text{nondim}} + \gamma_{i+1,\text{nondim}}) \end{aligned}$$

```
def coefficients():
    c1 = 0.0
    for i in range(npts-1):
        x1, y1 = Points[i,0], Points[i,1]
        x2, y2 = Points[i+1,0], Points[i+1,1]
        G1 = nondim_Gamma[i]
        G2 = nondim_Gamma[i+1]
        li = np.sqrt((x2-x1)**2 + (y2-y1)**2)
        c1 = c1 + li*(G1+G2)
    return c1
```