MAE 635 - Comp Fluid Dynamics II

Validation Simulation Supersonic Linearized Theory

Reference papers : *NASA Technical Paper 1406, US Standard Atmosphere*
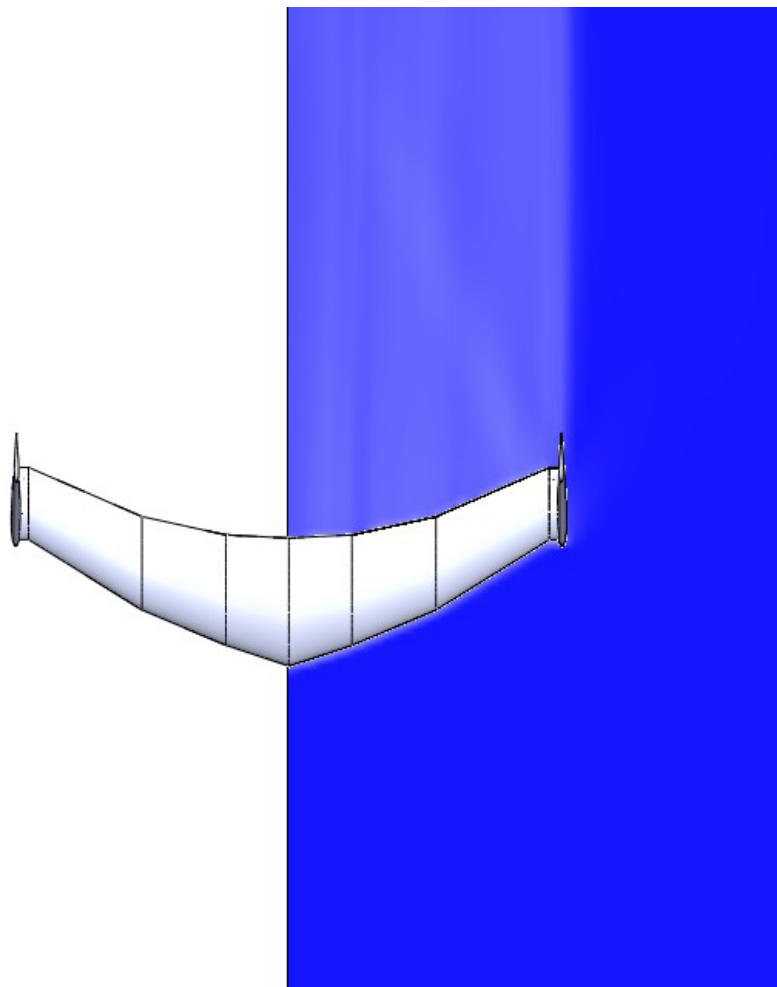
Abstract

This paper presents a comparative study of various techniques to compute the Mach angle on a slope under supersonic conditions. The approaches are analyzed and benchmarked against the methodologies described in NASA Technical Paper 1406, with an emphasis on accuracy and computational efficiency. By exploring traditional and modern numerical methods within a computational fluid dynamics (CFD) framework, the study aims to highlight the strengths and limitations of each technique in predicting shock phenomena in supersonic flows.

Contents

1 Theory

1.1 Motivation

The shock angle β is a key parameter in the analysis of supersonic flows, as it determines the region of influence of disturbances and the behavior of shock waves. The shock angle is defined as the angle between the shock wave and the direction of the incoming flow. It is related to the Mach number of the flow and the angle of attack of the object causing the shock wave.

$$M_{n1} = M_\infty \sin\beta \tag{1}$$

(a) Pressure behind the shock   *(b) Local pressure coefficient*

$$\frac{p_2}{p_\infty} = 1 + \frac{2\gamma}{\gamma+1}\Big(M_{n1}^2 - 1\Big), \qquad C_{p,2} = \frac{p_2 - p_\infty}{\frac{1}{2}\gamma p_\infty M_\infty^2} = \frac{2}{\gamma M_\infty^2}\Big(\frac{p_2}{p_\infty} - 1\Big).$$

So Knowing the Mach angle allows us to verify the validity of theory and the numerical method. The Mach angle is a key parameter in the analysis of supersonic flows, as it determines the region of influence of disturbances and the behavior of shock waves.

In the *small-disturbance limit* ($\theta \ll 1$, $\beta \approx \mu$) these formulas degenerate to the familiar linear results

$$C_p \approx \frac{2\theta}{\sqrt{M_\infty^2 - 1}}, \qquad C_L \approx -\frac{2\theta^2}{\sqrt{M_\infty^2 - 1}}, \qquad C_D \approx \frac{2\theta^2}{\sqrt{M_\infty^2 - 1}}.$$

1.2 Hypothesis

- Inviscid, adiabatic, perfect-gas flow, steady

$$P = \rho RT \quad ; \quad \gamma = const \quad ; \quad div(v) = 0 \quad ; \quad \frac{\partial}{\partial t} = 0$$

- Small perturbations of the flow field

$$v = v_0 + v' \quad ; \quad P = P_0 + P' \quad ; \quad T = T_0 + T'$$

- Disturbance velocities

$$v' = \epsilon v_1 + \epsilon^2 v_2 + \epsilon^3 v_3 + ... \quad ; \quad \epsilon << 1$$

1.3 Governing equations

The full compressible potential equation, which is a second-order partial differential equation, can be expressed in terms of the velocity potential ϕ as follows:

$$(1 - M^2\phi_x^2)\phi_{xx} - 2M^2\phi_x\phi_y\phi_{xy} + (1 - M^2\phi_y^2)\phi_{yy} + \phi_{zz} = 0, \tag{2}$$

where M is the Mach number, and ϕ_x, ϕ_y, and ϕ_z are the partial derivatives of the velocity potential with respect to the spatial coordinates x, y, and z, respectively. The subscripts denote partial differentiation.

1.4 Linearized equations

The linearized equations are obtained by substituting the perturbation velocities into the governing equations and neglecting higher-order terms. The resulting equations can be expressed as follows:

$$(1 - M_\infty^2)\phi_{xx} + \phi_{yy} + \phi_{zz} = 0. \tag{3}$$

This limited development neglects all terms of order $O(\phi_x^2)$ and above, as well as shock-shock interactions, thereby restricting valid Mach and deflection ranges.

1.5 Shock angle

1.5.1 Exact relation

The exact oblique-shock relation for a wedge half-angle θ is given by the following equation in the *NASA Technical Paper 1406*:

$$tan\theta = 2\cot\beta \frac{M_\infty^2 \sin^2\beta - 1}{M_\infty^2(\gamma + \cos 2\beta) + 2}. \tag{4}$$

This equation cannot be solved analytically for β in terms of θ and M_∞. In fact it's a transcendental equation. However, we can use the bisection method to find the solution numerically.

1.5.2 Numerical solution

As mentionned above the equation 1.5.1 is not analytically solvable for β. We can then use the bisection method to find the solution numerically. The bisection method is a root-finding algorithm that repeatedly bisects an interval and selects the subinterval in which a root exists. It is a simple and robust method for finding roots of continuous functions.

Therefore, you will find below the Python code that implements the bisection method to find the shock angle β for a given deflection angle θ and Mach number M_∞. The code is organized into several functions, each responsible for a specific task in the process of finding the shock angle.

- **Computation from β to θ:**
  The function `theta_from_beta(beta, M, gamma)` computes the deflection angle θ from the shock angle β using the exact oblique-shock relation:

$$\theta = \arctan\left(\frac{2\cot\beta\,(M^2\sin^2\beta - 1)}{M^2(\gamma + \cos 2\beta) + 2}\right).$$

```
1  def theta_from_beta(beta, M, gamma=1.4):
2          """
3          Returns theta (rad) for an oblique shock.
4          Accepts beta as scalar or numpy array.
5          """
6          beta = np.asarray(beta)
7          term1 = 2.0 / np.tan(beta)
8          term2 = M**2 * np.sin(beta)**2 - 1.0
9          term3 = M**2 * (gamma + np.cos(2.0 * beta)) + 2.0
10         return np.arctan(term1 * term2 / term3)
```

- **Bisection Method:**
  The function `bisection(beta_low, beta_high, f, tol, max_iter)` applies the bisection method to solve

$$f(\beta) = \theta(\beta) - \theta_{\text{target}} = 0,$$

where θ_{target} is the prescribed deflection angle. It refines the interval $[\beta_{\text{low}}, \beta_{\text{high}}]$ until the value of $f(\beta)$ is within a specified tolerance.

```
1  def bisection(beta_low, beta_high, f, tol=1e-10, max_iter=100):
2          """Find root of f(beta)=0 using bisection method (beta in rad)."""
3          f_low = f(beta_low)
4          for _ in range(max_iter):
5                  beta_mid = 0.5 * (beta_low + beta_high)
6                  f_mid = f(beta_mid)
7                  if abs(f_mid) < tol:
8                          return beta_mid
```

```
9                  if f_low * f_mid < 0.0:
10                       beta_high = beta_mid
11                  else:
12                       beta_low, f_low = beta_mid, f_mid
13        raise RuntimeError("Bisection_did_not_converge.")
```

- **Bracket Finding:**
The function `bracket_roots(theta_rad, M, gamma, n_scan)` scans a range of `beta` values—from the Mach angle $\sin^{-1}(1/M)$ up to nearly $90°$—to detect intervals where $f(\beta)$ changes sign. These intervals are the candidate regions where a root exists.

```
1  def bracket_roots(theta_rad, M, gamma=GAMMA, n_scan=4000):
2      """
3      Scan beta from Mach line to 89.9 deg:
4      returns list of intervals [beta_low, beta_high] where f changes sign.
5      """
6      beta_min = math.asin(1.0 / M) + 1e-6
7      beta_max = math.radians(89.9)
8
9      betas = np.linspace(beta_min, beta_max, n_scan)
10     f_vals = theta_from_beta(betas, M, gamma) - theta_rad
11
12     idx = np.where(np.diff(np.sign(f_vals)))[0]
13     return [(betas[i], betas[i + 1]) for i in idx]
```

- **Determining `beta` Solutions:**
The function `beta_solutions(theta_deg, M, gamma)` computes the shock angle(s) for a given deflection angle θ (converted from degrees to radians). It uses the previously identified bracketing intervals and applies the bisection method within each interval to accurately compute the corresponding `beta` values. Depending on the case, it may return one or two solutions (typically corresponding to the weak and strong shock cases).

```
1  def beta_solutions(theta_deg, M, gamma=GAMMA):
2      """
3      Returns a list [beta_weak (rad), beta_strong (rad) (optional)]
4      for a given (theta, M) pair. 0 or 1 or 2 solutions.
5      """
6      theta_rad = math.radians(theta_deg)
7      intervals = bracket_roots(theta_rad, M, gamma)
8      betas = []
9      for beta_low, beta_high in intervals:
10         betas.append(
11             bisection(beta_low, beta_high,
12                       lambda b: theta_from_beta(b, M, gamma)
13                       - theta_rad))
14     return betas
```

With the code below it is possible to solve β for arrays of θ and M_∞ values. By doing so and plotting the results, we can compare the numerical solution with the analytical solution given by the code with a reviewed version from [**?**].
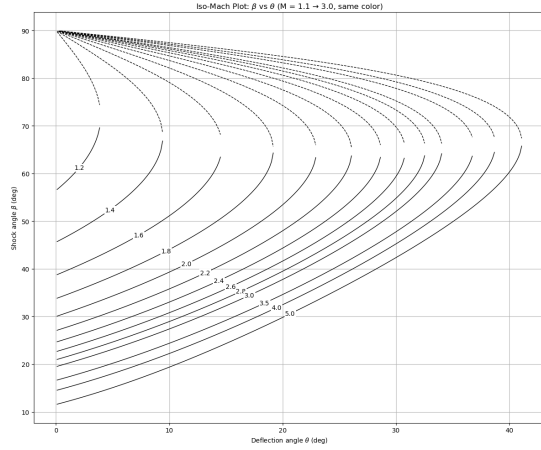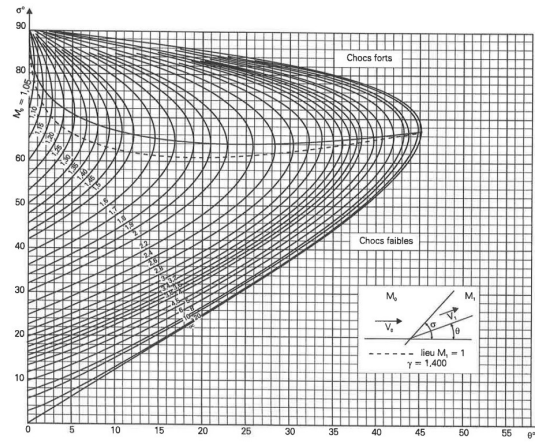
Figure 1: Computed polar

Figure 2: Supaero polar

We can see that the results are similar showing this resolution is valid. The difference between the two results is due to the fact that the code from Supaero is using a diffrent type of plotting.

1.5.3 Linearized shock angle

Because the formal solution is complex to compute, we can use the small-disturbance approximation to obtain a linearized version of the shock angle. In the limit $\theta \to 0$ with attached flow, the critical condition arises when the numerator vanishes:

$$M_\infty^2 \sin^2 \beta_d - 1 \approx 0 \tag{5}$$

$$\implies \quad \beta_d \approx \sin^{-1}\left(\frac{1}{M_\infty}\right) \tag{6}$$

Then the equation below offers a first-order shock-detachment estimate. Note the distinction from the Mach angle $\mu = \sin^{-1}(1/M_\infty)$, which applies to infinitesimal disturbances rather than finite shocks.

Linearised ShockWaves