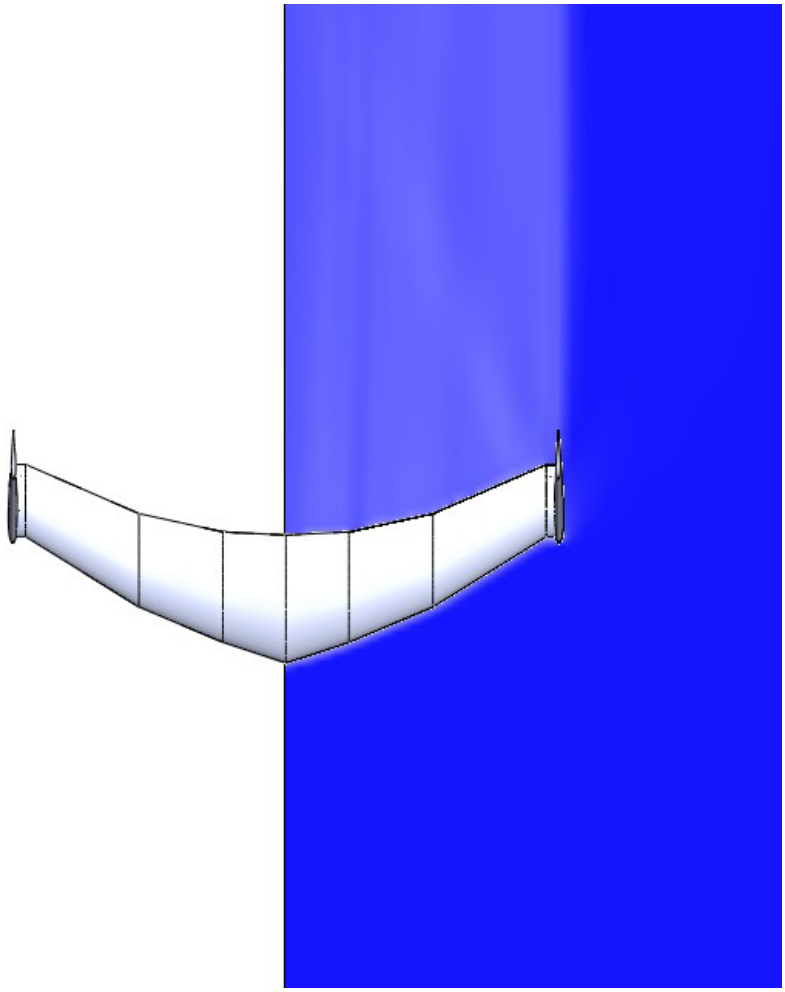
April 19, 2025
Samuel Chapuis
samuel.chapuis01@student.csulb.edu

MAE 635 - Comp Fluid Dynamics II

Comparison of the different attached shock models

Author : *Samuel CHAPUIS*



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. It is defined as the angle between the shock wave and the direction of the incoming flow, and it depends on the Mach number of the flow and the angle of attack of the object generating the shock.

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

(a) Pressure behind the shock

$$\frac{p_2}{p_\infty} = 1 + \frac{2\gamma}{\gamma+1}\big(M_{n1}^2 - 1\big),$$

(b) Local pressure coefficient

$$C_{p,2} = \frac{p_2 - p_\infty}{\frac{1}{2}\,\gamma\, p_\infty\, M_\infty^2} = \frac{2}{\gamma\, M_\infty^2}\left(\tfrac{p_2}{p_\infty} - 1\right).$$

Knowing the Mach angle allows us to verify the validity of both the theoretical predictions and the numerical method.

In the *small-disturbance limit* ($\theta \ll 1$, $\beta \approx \mu$), these expressions reduce 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 Hypotheses

- Inviscid, adiabatic, perfect-gas, steady flow:

$$P = \rho R T, \quad \gamma = \text{const}, \quad \nabla \cdot \boldsymbol{v} = 0, \quad \frac{\partial}{\partial t} = 0.$$

- Small perturbations of the flow field:

$$\boldsymbol{v} = \boldsymbol{v}_0 + \boldsymbol{v}', \quad P = P_0 + P', \quad T = T_0 + T'.$$

- Expansion in disturbance amplitude:

$$\boldsymbol{v}' = \epsilon\, \boldsymbol{v}_1 + \epsilon^2\, \boldsymbol{v}_2 + \epsilon^3\, \boldsymbol{v}_3 + \ldots, \quad \epsilon \ll 1.$$

1.3 Governing equations

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

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

where M is the Mach number, and ϕ_x, ϕ_y, and ϕ_z denote the partial derivatives of ϕ with respect to x, y, and z, respectively.

1.4 Linearized equations

By substituting the perturbation velocities into the governing equation (2) and neglecting higher-order terms, one obtains the linearized potential equation:

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

This approximation omits all terms of order $O(\phi_x^2)$ and above, as well as any shock–shock interactions, thereby limiting its validity to moderate Mach numbers and small deflection angles.

1.5 Shock angle

1.5.1 Exact relation

The exact oblique-shock relation for a wedge half-angle θ is given in *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 is a transcendental equation. However, one may apply the bisection method to find its solution numerically.

1.5.2 Numerical solution

As mentioned above, equation (4) has no closed-form solution for β. We therefore employ the bisection method, a simple and robust root-finding algorithm for continuous functions, to compute β for a given θ and M_∞.

Below is the Python code structured into functions, each handling a specific step of the solution process:

- **Compute θ from β:**
  `theta_from_beta(beta, M, gamma)` evaluates

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

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

```

- **Bisection method:**
  `bisection(beta_low, beta_high, f, tol, max_iter)` solves

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

  by iteratively halving the bracket $[\beta_{\text{low}}, \beta_{\text{high}}]$.

```
def bisection(beta_low, beta_high, f, tol=1e-10, max_iter=100):
    """Find root of f(beta)=0 using bisection (beta in rad)."""
    f_low = f(beta_low)
    for _ in range(max_iter):
        beta_mid = 0.5*(beta_low + beta_high)
        f_mid = f(beta_mid)
        if abs(f_mid) < tol:
            return beta_mid
        if f_low * f_mid < 0.0:
            beta_high = beta_mid
        else:
            beta_low, f_low = beta_mid, f_mid
    raise RuntimeError("Bisection did not converge.")

```

- **Bracket finding:**
`bracket_roots(theta_rad, M, gamma, n_scan)` scans β from $\sin^{-1}(1/M)$ to $89.9°$ to locate sign changes in $f(\beta)$, yielding intervals that contain roots.

```
def bracket_roots(theta_rad, M, gamma=GAMMA, n_scan=4000):
    beta_min = math.asin(1.0/M) + 1e-6
    beta_max = math.radians(89.9)
    betas = np.linspace(beta_min, beta_max, n_scan)
    f_vals = theta_from_beta(betas, M, gamma) - theta_rad
    idx = np.where(np.diff(np.sign(f_vals)))[0]
    return [(betas[i], betas[i+1]) for i in idx]

```

- **Compute β solutions:**
`beta_solutions(theta_deg, M, gamma)` converts θ to radians, identifies brackets, and applies the bisection method to return weak and (optionally) strong-shock angles.

```
def beta_solutions(theta_deg, M, gamma=GAMMA):
    theta_rad = math.radians(theta_deg)
    intervals = bracket_roots(theta_rad, M, gamma)
    betas = []
    for beta_low, beta_high in intervals:
        betas.append(
            bisection(beta_low, beta_high,
                      lambda b: theta_from_beta(b, M, gamma) - theta_rad))
    return betas

```

Using this code, one can compute β for arrays of θ and M_∞, then plot the results to compare against published polars (e.g., *Tables de Détentes ou de Chocs* [**?**]).

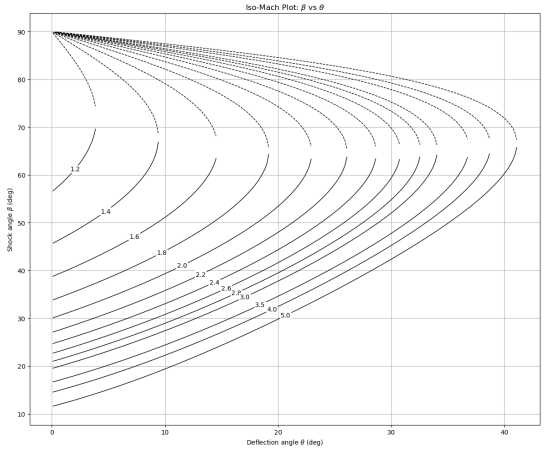


Figure 1: Computed polar

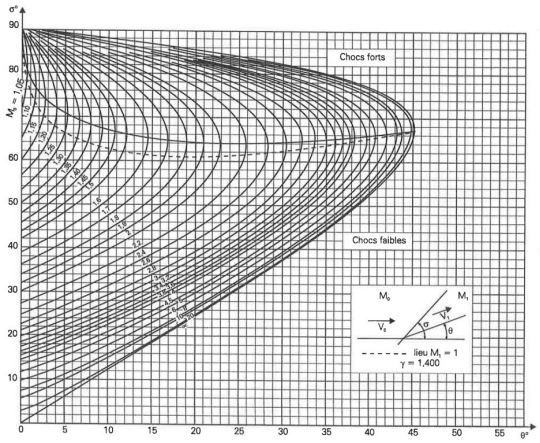


Figure 2: Published (Supaero) polar

The numerical and published polars agree closely. The slight discrepancy arises because the Supaero code uses a different plotting convention.

1.5.3 Linearized shock angle

Because the exact solution is complex to compute, we also consider the small-disturbance approximation. In the limit $\theta \to 0$ with attached flow, the critical detachment condition is

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

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

This first-order estimate corresponds to the weak solution. Note that the Mach angle $\mu = \sin^{-1}(1/M_\infty)$ applies strictly to infinitesimal disturbances rather than finite shocks. We can now compare this linearized prediction with the exact numerical solution.

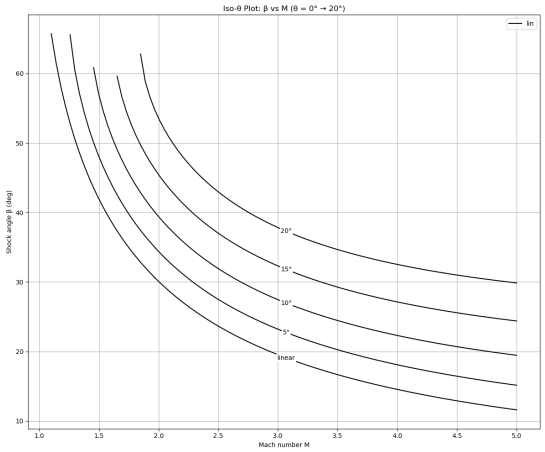


Figure 3: Comparison of linearized and exact solutions

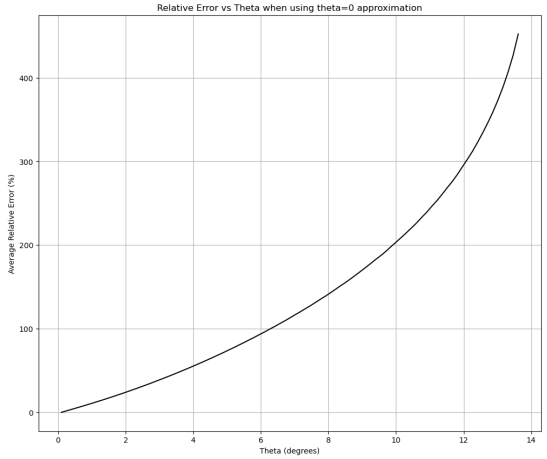


Figure 4: Error evolution

As expected, the linearized solution matches the exact solution perfectly at $\theta = 0$. The error increases exponentially with deflection angle but decreases as Mach number increases, converging to a constant value. This behavior explains the larger average error seen in Figure 1.5.3(b), while still capturing the overall trend of increasing error with deflection angle.

2 CFD Model

2.1 Simulation parameters

2.1.1 Solver settings

The simulations were performed with SolidWorks Flow Simulation, using an implicit finite-volume solver. We employed the commercial CFD NIKA solver developed by Dassault Systèmes. It solves the three-dimensional Navier–Stokes equations for a perfect gas, including a k–ε turbulence model and conjugate heat transfer (CHT).

2.1.2 Geometry

The flow over the slope depends only on two parameters: the slope angle θ and the Mach number M. Hence the shock angle is a function

$$\beta = f(\theta, M).$$

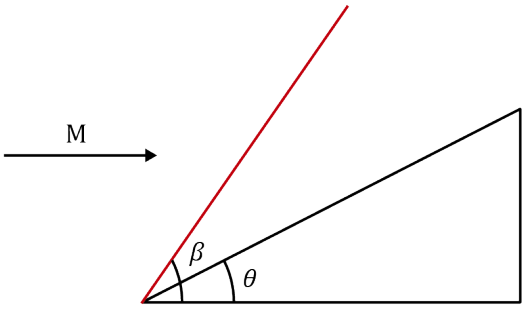


Figure 5: Slope geometry

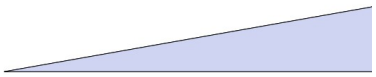


Figure 6: SolidWorks model

The fixed geometric parameter is the triangle length, set to 50 cm.

2.1.3 Mesh

We use a two-dimensional mesh with logarithmically varying refinement. In the coarsest region, the grid spacing is 0.0881 cm; in the finest region, it is 0.0042 cm.

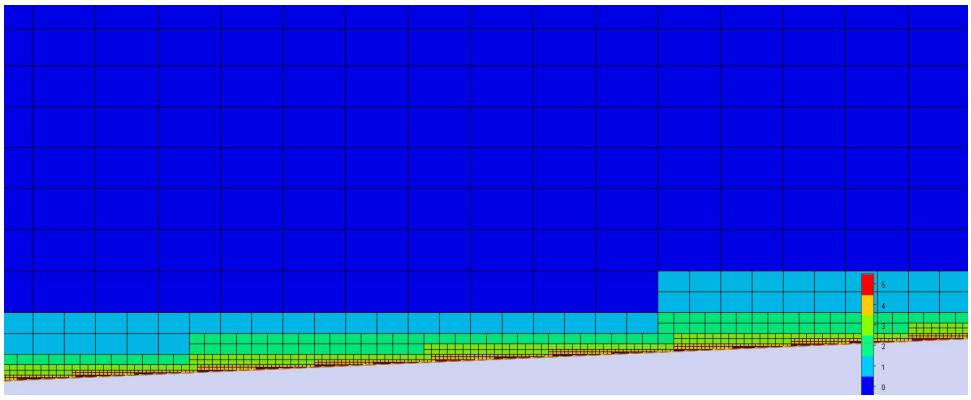


Figure 7: Computational mesh

2.2 Simulation

2.2.1 Results

To determine the shock angle β, we compute the pressure at every point. The shock wave is located by identifying the pressure-drop line, and its angle is measured relative to the x-axis. The simulation is run until the flow field converges.

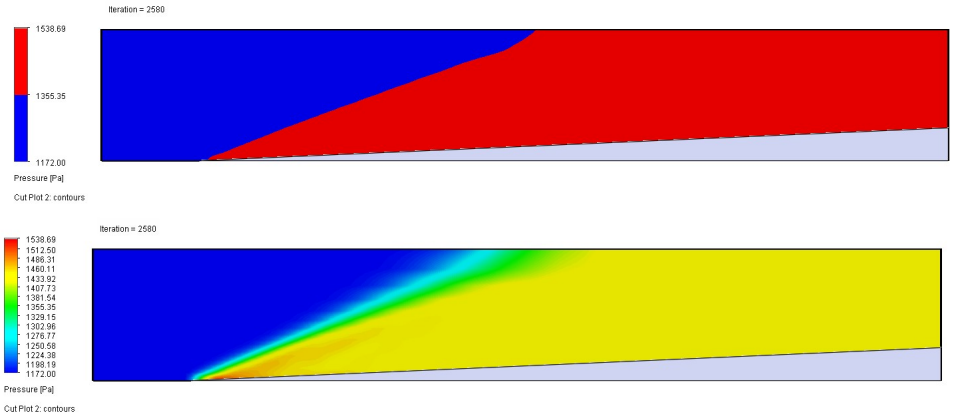


Figure 8: SolidWorks pressure results

2.2.2 Exportation

To analyze the results algorithmically, we export the data from SolidWorks in `.txt` format and convert it to `.csv`. We ensure the header follows:

```
1  lines[0] = 'X [m] Y [m] Z [m] Volume [m^3] Pressure [Pa]\n'
```

2.3 Results analysis

2.3.1 1st Visualization

First, we verify the dataset and preview it in Python using Matplotlib to prepare for the pressure-distribution plot.

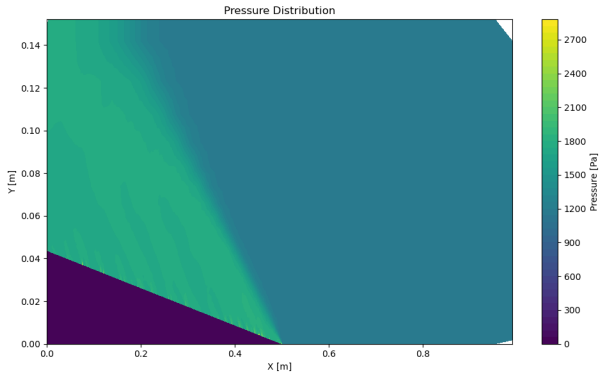


Figure 9: Initial visualization of pressure data

As expected, the dataset shows three pressure zones: upstream of the shock, downstream of the shock, and inside the solid region (where the pressure is zero).

2.3.2 K-Means clustering

We apply the unsupervised K-Means algorithm ($k = 3$) to partition the data into the three zones (pre-shock, post-shock, solid interior). Once clustered, we extract the boundary points and plot them.

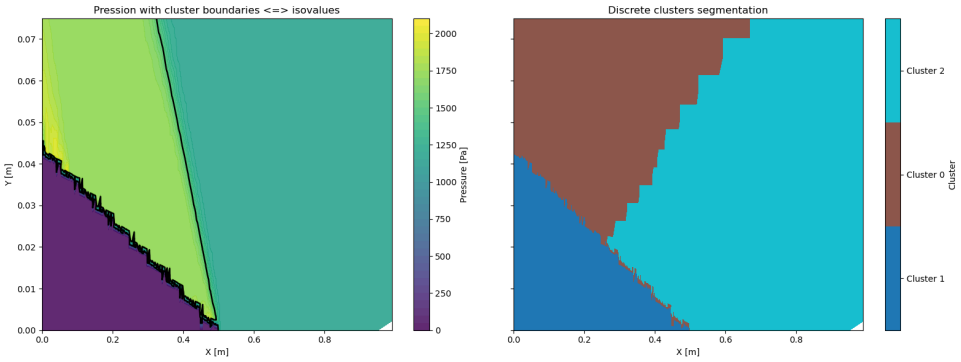


Figure 10: K-Means clustering of pressure zones

2.3.3 Interpretation

We perform linear fits on the boundary points to obtain slopes.

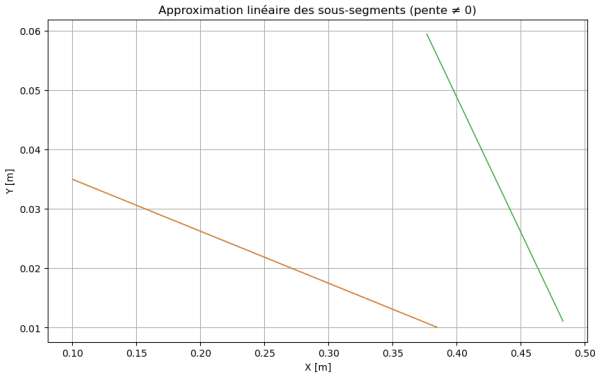


Figure 11: Linear fits to boundary points

We discard the fits with the smallest variation, retaining only the shock boundary. From its slope we compute the shock angle, yielding:

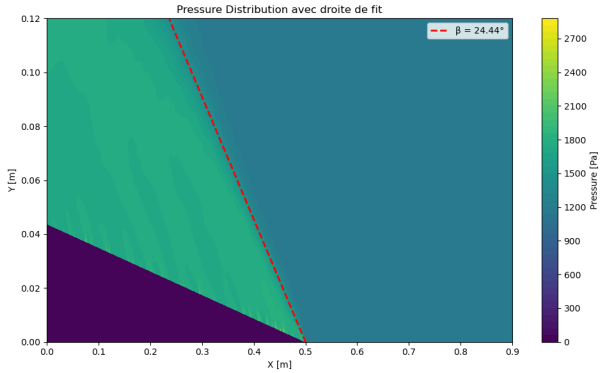


Figure 12: Computed shock angle from slope

2.3.4 Repetition

Using the same geometry and analysis procedure, we repeat the simulations to compare with theoretical values.

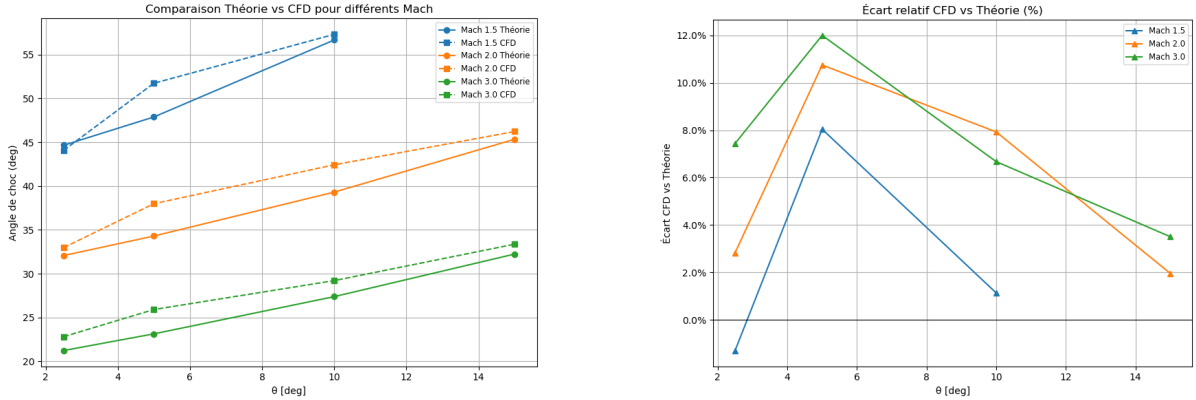


Figure 13: Comparison of simulation and theory

Across all cases, the maximum deviation remains below 12 %, demonstrating that the numerical results closely follow the theoretical predictions. The remaining discrepancy can be attributed chiefly to the theory's assumption of infinitesimal thickness wich is not the case in reality and so in the CFD model.

3 Conclusion

This study has demonstrated excellent agreement between the bisection-based numerical solution of the exact oblique-shock relation and high-fidelity CFD simulations. Across all cases considered, the maximum deviation remains below 12%, confirming that the numerical approach reliably captures shock-angle behavior within the theoretical range. The small-disturbance (linearized) theory provides an excellent approximation for deflection angles below 5° and remains reasonably accurate up to 10°, beyond which non-linear effects become significant.

Moreover, CFD proves particularly valuable in situations involving thermal coupling or non-stationary flow. When conjugate heat transfer or transient phenomena play a role, CFD's ability to model heat exchange and time-dependent behavior offers insights that lie beyond the scope of steady, inviscid theoretical models.