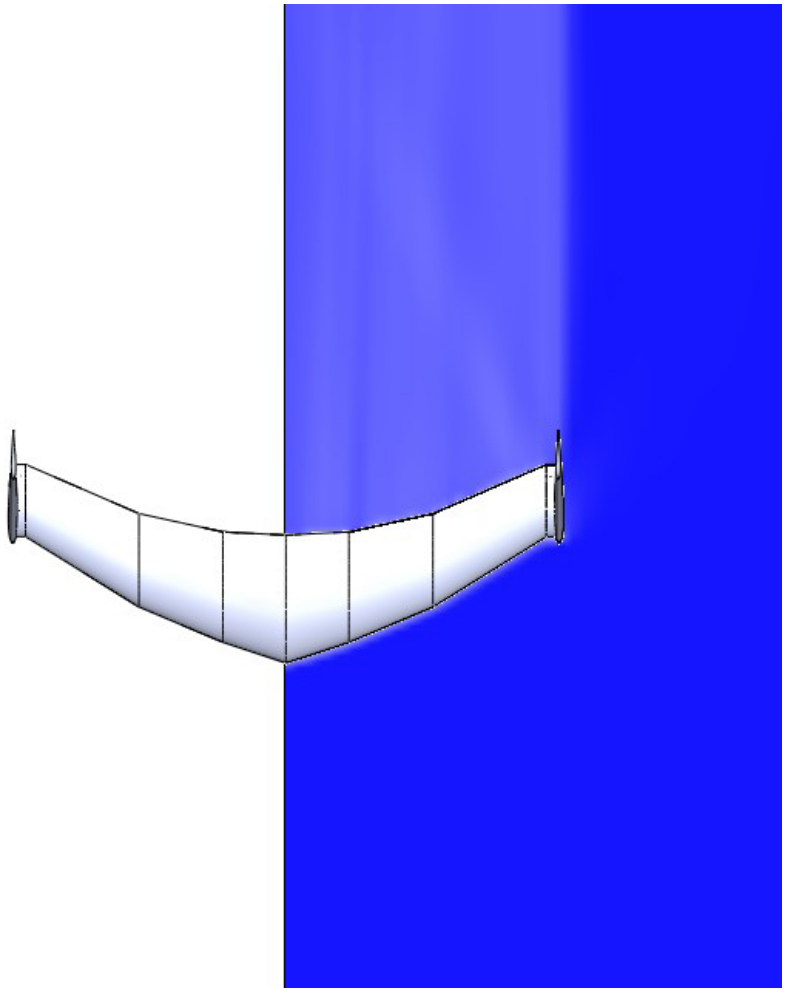
April 19, 2025
Samuel Chapuis
samuel.chapuis01@student.csulb.edu

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 $\beta$ 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*

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

*(b) Local pressure coefficient*

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

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 $\phi$ 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 $\phi_x$, $\phi_y$, and $\phi_z$ 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 $\theta$ 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 $\beta$ in terms of $\theta$ and $M_\infty$.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 $\beta$. 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 $\beta$ for a given deflection angle $\theta$ and Mach number $M_\infty$. The code is organized into several functions, each responsible for a specific task in the process of finding the shock angle.

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

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

```
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:**
  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 $\theta_{\text{target}}$ 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.

```
def bisection(beta_low, beta_high, f, tol=1e-10, max_iter=100):
    """Find root of f(beta)=0 using bisection method (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:**
  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.

```
def bracket_roots(theta_rad, M, gamma=GAMMA, n_scan=4000):
    """
    Scan beta from Mach line to 89.9deg:
    returns list of intervals [beta_low, beta_high] where f changes sign.
    """
    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]

```

- **Determining `beta` Solutions:**
  The function `beta_solutions(theta_deg, M, gamma)` computes the shock angle(s) for a given deflection angle $\theta$ (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).

```
def beta_solutions(theta_deg, M, gamma=GAMMA):
    """
    Returns a list [beta_weak (rad), beta_strong (rad) (optional)]
    for a given (theta, M) pair. 0 or 1 or 2 solutions.
    """
    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

```

With the code below it is possible to solve $\beta$ for arrays of $\theta$ and $M_\infty$ 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 *Tables de Détentes ou de Chocs*. [**?**]

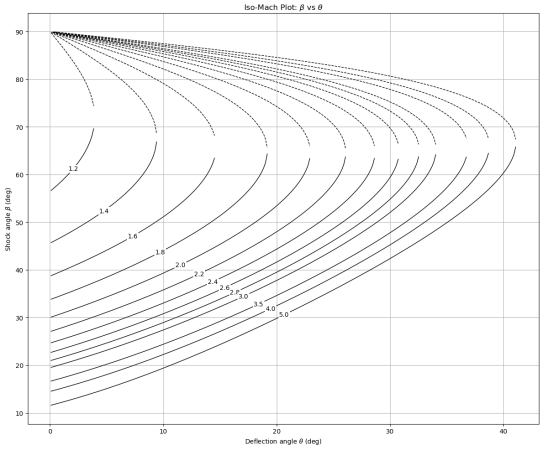



Figure 1: Computed polar

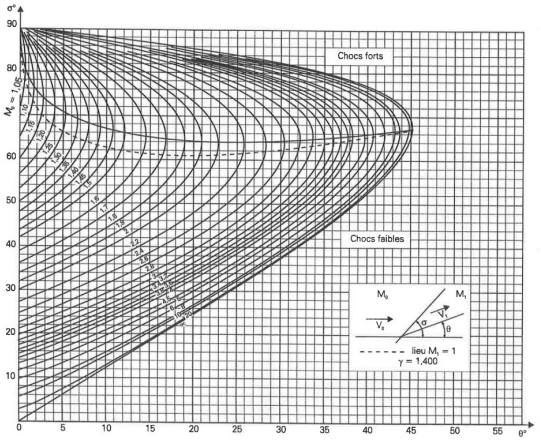



Figure 2: Supaero polar

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:

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

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

Then the equation below offers a first-order shock-detachment estimate corresponding to the weak solution. Note the distinction from the Mach angle $\mu = \sin^{-1}(1/M_\infty)$, which applies to infinitesimal disturbances rather than finite shocks. Knowing this we can now compute the shock angle for a given deflection angle and Mach number with this approximation and compare it with the exact solution.

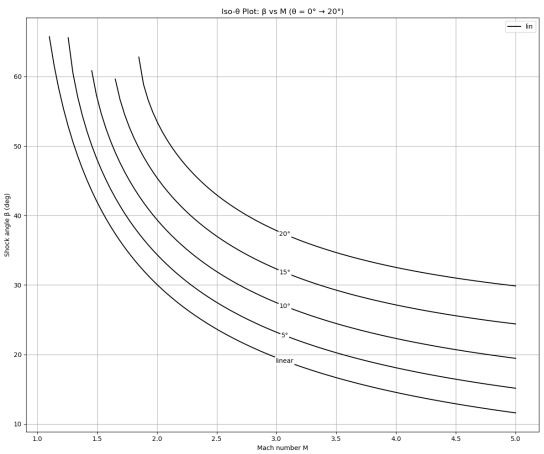



Figure 3: Comparaison linearized arithmetical

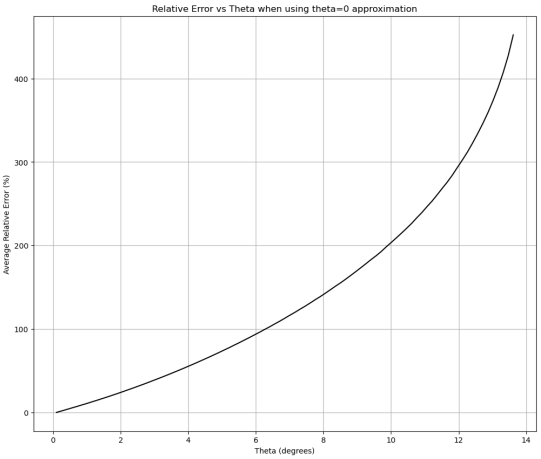



Figure 4: Evolution of the error

As expected the solution is perfectly matching the exact solution for $\theta = 0$. We can see that the error

is not linear and is increasing exponentialy with the deflection angle. We can also see that the error is decreasing with the Mach number to converge to a constant error value. This explain the colossal average error on the figure 1.5.3 (b). However, it still shows the general trend of error increase with the deflection angle.

# 2 CFD Model

## 2.1 Simulation parameters

### 2.1.1 Solver settings

The simulation is made with SolidWorks Flow Simulation. It use an implicit solver based on finit volumses (FVM). The solver is CFD NIKA which is a commercial solver create by Dassault Systems. It's base on a 3d flow Navier-Stocks equations with a turbulent model $k - \epsilon$ and conjugate heat transfer (CHT).

### 2.1.2 Geometry

The flow on a slope is depending only of two dimensions. So the problem is reduced to only two varialb the slope $\theta$ and the Mach number $M$. So that the solution is $\beta = f(\theta, M)$.

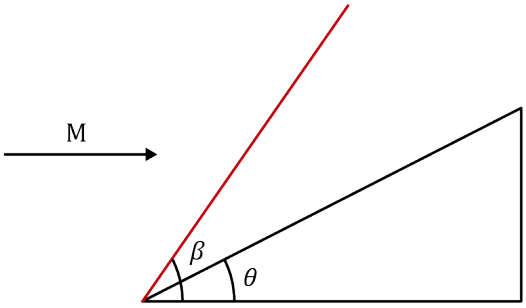



Figure 5: Geometry

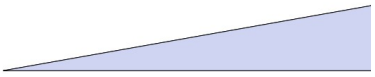


Figure 6: SolidWorks View

The fix variable of the geometry is here the length of the triangle fix at $50cm$.

### 2.1.3 Mesh

For the problem we can use a 2D mesh with a refinement that progress logaritmically whith the distance. At the largest part of the grid there is a point evry $0.0881cm$ and at the thinest part of the grid $0.0042cm$.

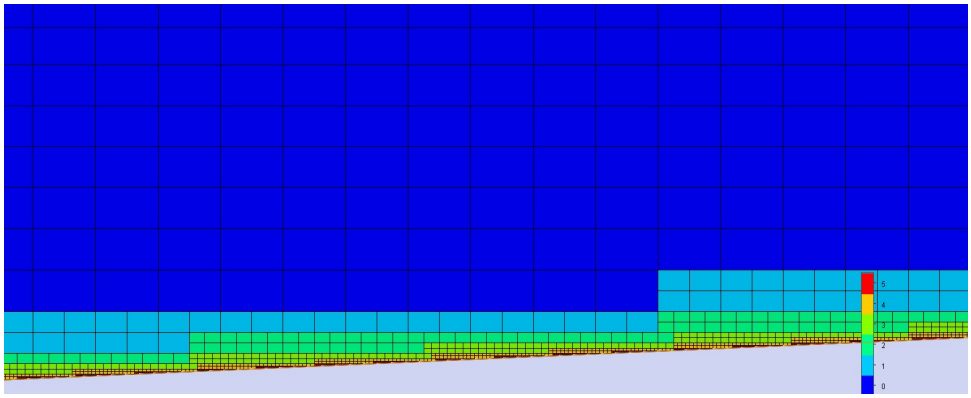


Figure 7: Mesh

## 2.2 Simulation

### 2.2.1 Resutls

In the objective to get the shock angle $\beta$ the best solution is to compute the pressure in evry points. So the shock wave will be place where there is a pressure drop line, then compute the angle between this line and the x axe. To do the simulation run until a flow-mas convergent solution is found.

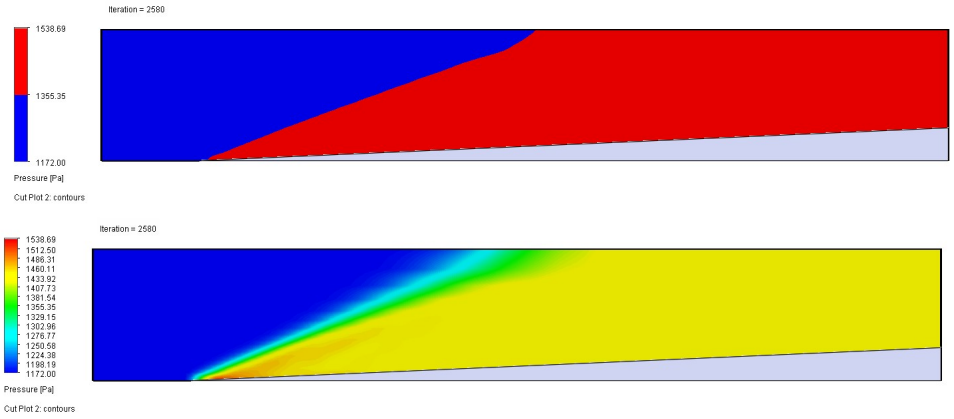



Figure 8: SolidWorks Pressure Results

### 2.2.2 Exportation

To be able to analys the results it is needed to export them out of SolidWorks because we can't apply algorithm to find the shock angle inside.
The exportation are on the .txt format wich is hard to read so it was important to convert this into .csv. On top of that it's important to make sure that the varaible are as following

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

## 2.3 Results analysis

### 2.3.1 $1^{st}$ Visualization

To make sure that we are working on the right dataset and also to plot later the result on the pressure chart. It is important first to see the data inside python, trough matplotlib here.

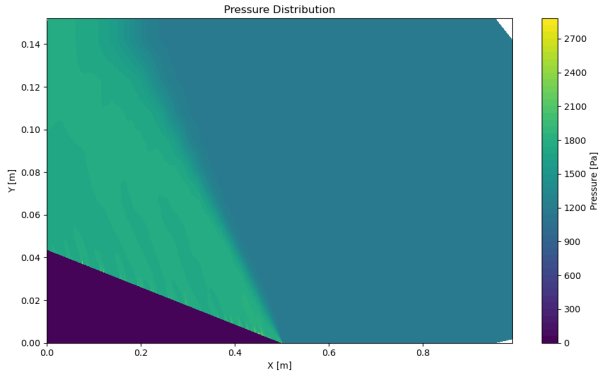



Figure 9: $1^{st}$Visualization

As expected we have 3 pressure zone in the dataset. One before the shock, one after the shock a not a real on insinde the geometry where the pressure is 0 because it's inside the solide.

### 2.3.2 K-Means

To manage to find the three pressure zones explained before — the region before the shock, the region after the shock, and the interior of the solid — we use an unsupervised machine learning algorithm called K-Means.

K-Means is a clustering algorithm that partitions a set of data points into a predefined number of clusters (in our case, $k = 3$). The goal is to group the data so that points within each cluster are as close as possible to the cluster center (centroid), while being as far as possible from the centers of the other clusters.

Once clustered, we can export the points that compose the borders and plot them.

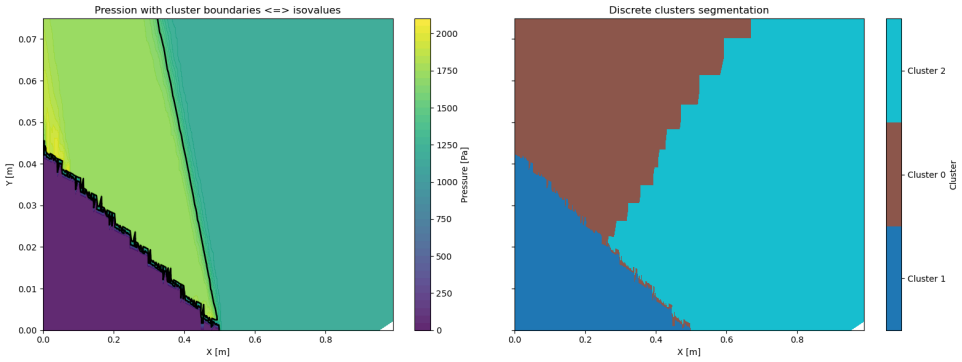



Figure 10: Clustering

### 2.3.3 Interpetation

Now it is possible to do some linear approximation on the borders to get draw some slopes.

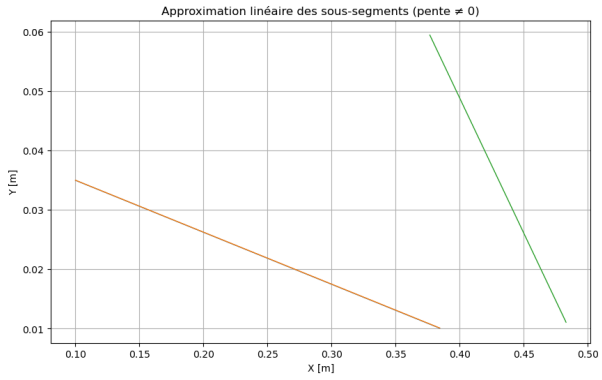



Figure 11: Slopes

With this slopes it is possible to reject the one with the smallest variation. Only to conserve the shock one and then compute the angle base on the slope, wich give the following result.

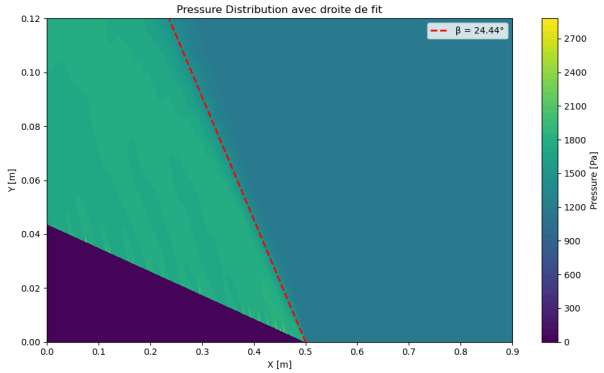



Figure 12: Simulation Interpetation

### 2.3.4 Repetition

With the geometry parameters and the process to analys the result it is possible to duplicate the simulations to compare with the theoritical values.

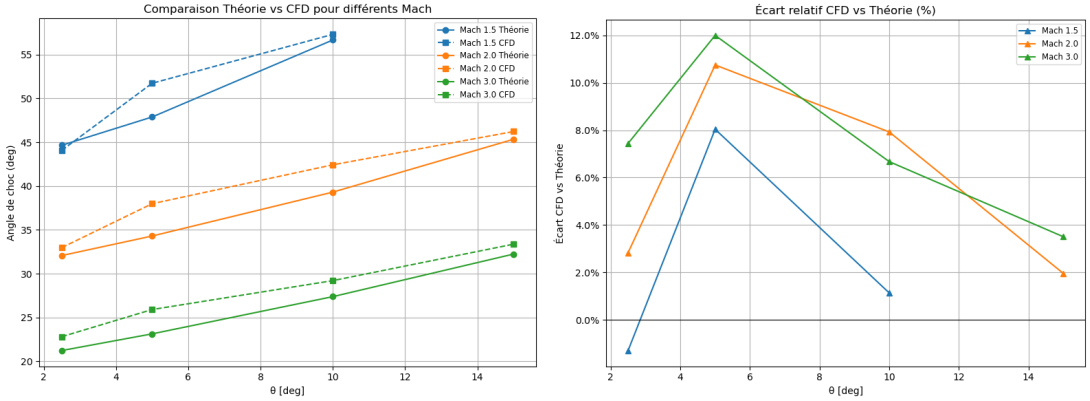



Figure 13: Solution comparaison

We can see the error is never over 12% so it is clear that the simulation feat well with the theory. This error is probably mainly due to the hypothesis of infinitesimal thickness in the theory.

# 3 Conclusion

To conlcude it is clear that the simulation and the numerical solution are extremely close. So in the range of the theory it is clear that the numerical solution is largely enough to understand the actual phenomena. In addition to that, the linearized theory is sufficient for small angles $< 5^\circ$ and decent for angles $< 10^\circ$.