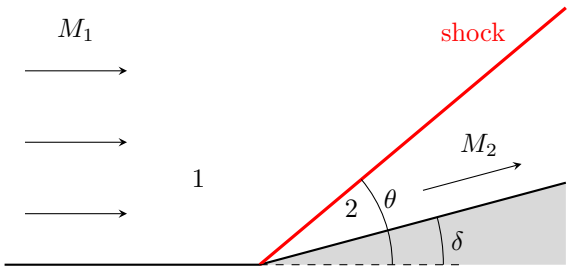
ASEN 3502 Computational Methods
Lab 2: Nonlinear Equations

Exact Due Date on Gradescope

General Instructions

1. You may use any programming language, and you may use different languages for different parts. However, the primary goal of this lab is understanding, and *you should use Matlab unless every team member is enthusiastic about using something else.*
2. Adhere to the AI policy at asen3502.com/ai-policy.

In a gas, pressure information travels at the speed of sound. When an aircraft flies faster than the speed of sound, information cannot flow upstream to smoothly divert the flow. This means that the airflow must change direction near instantaneously to move around the aircraft. This abrupt deflection and compression is known as a shock. In this lab, we will solve nonlinear equations that determine the angle of oblique shocks.



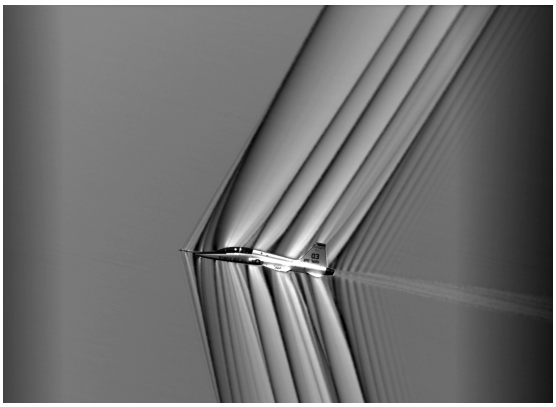


Figure 1: Left: Oblique shock on a wedge of angle δ. Right: schlieren image of a T-38 in supersonic flight (NASA).

Supersonic flow at Mach number M_1 over a wedge of angle δ (Figure 1) produces an oblique shock at angle θ given by NACA Report 1135, Eq. 139b:

$$\tan\delta = \frac{M_1^2 \sin 2\theta - 2\cot\theta}{2 + M_1^2\,(\gamma + \cos 2\theta)}. \tag{1}$$

This equation does not have a simple analytic solution for θ, so we will solve it using numerical methods. For sufficiently small δ, the equation has two solutions, the smaller "weak" solution usually encountered in the real world, and the higher "strong" solution. If δ is too large, an oblique shock is impossible and a normal shock forms in front of the wedge, invalidating Equation (1). Throughout this lab, the ratio of specific heats, $\gamma = c_p/c_v$, is 1.4, and the Mach angle, μ, is $\sin^{-1}\left(\frac{1}{M}\right)$. Notation follows NACA Report 1135.

Compiled 2026-09-18 04:15:42

Deliverables (at end of Week 2)

1. To the **Lab 2 Writeup** assignment on Gradescope: A **single pdf** writeup that contains all of the items marked writeup.
   - Make sure to clearly label all plots and figures and use correct units where appropriate.
   - The lab writeup does *not* need to include narrative text unless specifically requested; it can just include the plot or results from each writeup task labeled with which task they are from.
2. To the **Lab 2 Code** assignment on Gradescope: Your **source code** for all code items.
   - Please help the TAs by clearly naming your files.
   - In this lab, you will *not* be graded based on documentation or comments in your code. (Though in practice, documentation is extremely important!)

1 Week 1: Oblique Shock Angle

1.1 Define the model

writeup code

Define a residual function $f(\theta)$ that will be zero at the solutions of Equation (1). Implement this as a function `shock_residual(delta, theta, M)`. Plot $f(\theta)$ for $M_1 = 3$ and $\delta = 10°, 20°, 30°, 40°$ on the interval $\mu \leq \theta \leq 90°$. As in all plots, make sure to label all axes and provide a legend when appropriate. Identify the weak root, the strong root, and the case with no root and note these in the writeup.

1.2 Derive and verify the derivative

writeup code

Derive $f'(\theta)$ analytically (Hint: define N and D so that $f(\theta) = N/D - \tan\delta$ and use the quotient rule) and implement it in `shock_derivative(delta, theta, M)`. Write a script `verify_derivative` that compares to a finite difference approximation

$$f'(\theta) \approx \frac{f(\theta + h) - f(\theta)}{h}$$

at several values of θ and δ. Choose h so that it is very small, but not small enough to cause roundoff errors.

1.3 Implement root-finding algorithms

code

Implement a subset of the following root-finding methods:

(a) incremental search
(b) bisection
(c) Newton-Raphson
(d) secant
(e) Brent's method (advanced)

Your team must implement **at least four of the methods**, and **each team member must implement at least one**. All of the methods should use the interface

```
[x, info] = method_name(f, fprime, interval, atol, maxit)
```

where `f` and `fprime` are handles for functions that accept one argument, `interval` is a two-element vector, and `x` is a root estimate within absolute tolerance (E_s) `atol` of a true root. `info` is a struct for auxiliary output; use `info = struct();` for now. Some methods will ignore or reinterpret some arguments.

1.4 Verification script

code

Write a script `verify_root_find` that tests every method on a function with a known root. It should use a subfunction `test_root_find` that takes a solver function handle and causes an error via `assert` if the solve fails. Call `test_root_find` with each of the solve functions.

1.5 Solve for the shock angle

writeup code

For $M_1 = 3$, $\delta = 20°$, find the weak and strong shock angles to within 10^{-6} degrees with every method. Mark your solutions on the θ vs δ Chart 2 in NACA 1135 (Recommendation: use screenshots of the pdf). Note that your solvers expect functions with single arguments. You can create these as anonymous functions, e.g.

```
@(theta) shock_residual(deg2rad(20), theta, 3)
```

1.6 Convergence study

writeup code

Add two fields to the `info` struct returned by each solver: `info.history.funcCount`, a vector with the cumulative number of `f` and `fprime` calls after each iteration, and `info.history.x`, a vector with the best root estimate after each iteration. For the weak shock in Task 1.5, plot the error in the estimate versus `funcCount` for all methods on a log scale. If one method needs drastically more calls than the others, plot it on its own figure.

1.7 Initial guess dependence

writeup

For Newton-Raphson, sweep the initial guess across $\mu \leq \theta_0 \leq 90°$ and record whether each guess converges to the weak root, the strong root, or fails. Present the result as a plot or table.

Compiled 2026-09-18 04:15:42

2 Week 2: Intersecting Shocks

Managing shocks in intakes in supersonic flight is a key challenge for high performance aircraft. The F-15 used a complex variable-geometry intake to achieve its Mach 2.5 top speed. Figure 2 shows a supersonic intake. Shocks A and B from the ramp (δ_A) and cowl lip (δ_B) intersect and continue as refracted shocks C and D. Regions 4 and 4′ are separated by a slip line at angle ϕ across which pressure and flow direction are equal but Mach number is not.

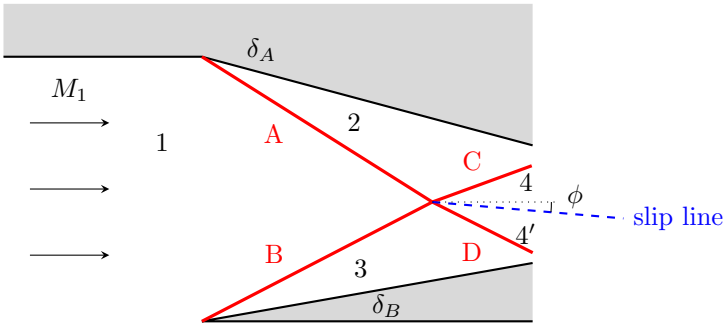


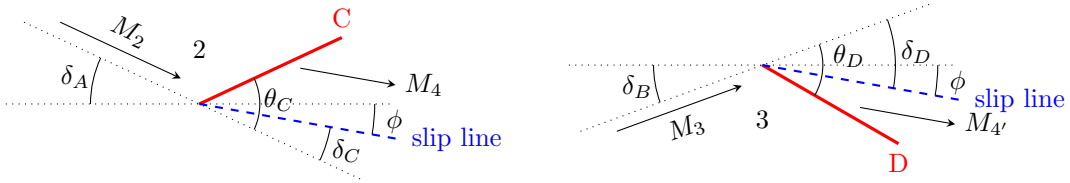
Figure 2: Intersection of shocks of opposite families (after Anderson Fig. 9.21). Right: the rectangular ramp inlets of an F-15 (USAF).

For a generic shock from upstream state u to downstream state d, NACA 1135 Eqs. 128, 131 give

$$\frac{p_d}{p_u} = \frac{2\gamma M_u^2 \sin^2\theta - (\gamma-1)}{\gamma+1}, \tag{2}$$

$$M_d^2 \sin^2(\theta-\delta) = \frac{(\gamma-1)M_u^2 \sin^2\theta + 2}{2\gamma M_u^2 \sin^2\theta - (\gamma-1)}. \tag{3}$$

The incident shocks are solved with Week 1 tools: θ_A from (M_1, δ_A) and θ_B from (M_1, δ_B), then p_2/p_1, M_2, p_3/p_1, M_3 from Eqs. (2) and (3). With ϕ taken positive when the slip line tilts toward the cowl, as drawn, the refracted shocks deflect the flow by $\delta_C = \delta_A - \phi$ and $\delta_D = \delta_B + \phi$ as shown below:



The three unknowns ϕ, θ_C, θ_D satisfy the two shock-angle relations

$$\tan\delta_C = \frac{M_2^2 \sin 2\theta_C - 2\cot\theta_C}{2 + M_2^2(\gamma + \cos 2\theta_C)}, \quad \tan\delta_D = \frac{M_3^2 \sin 2\theta_D - 2\cot\theta_D}{2 + M_3^2(\gamma + \cos 2\theta_D)}, \tag{4}$$

and pressure continuity across the slip line,

$$\frac{p_2}{p_1}\frac{p_4}{p_2} = \frac{p_3}{p_1}\frac{p_{4'}}{p_3}, \tag{5}$$

where p_4/p_2 is Eq. (2) evaluated with (M_2, θ_C) and $p_{4'}/p_3$ with (M_3, θ_D).

We will solve for the specific case where

$$M_1 = 3, \quad \delta_A = 15^\circ, \quad \delta_B = 10^\circ.$$

Compiled 2026-09-18 04:15:42

2.1 Define the model

writeup

Define a vector-valued residual function, $\{f\}(\{x\})$ where

$$\{x\} = \begin{Bmatrix} \phi \\ \theta_C \\ \theta_D \end{Bmatrix}$$

and $\{f\}(\{x\}) = \{0\}$ when Equations 4 and 5 are satisfied.

2.2 Newton-Raphson for systems

writeup

Write pseudocode for Newton-Raphson applied to $\{f\}(\{x\}) = \{0\}$ with Jacobian $[J](\{x\})$, including the linear solve $[J]\{\Delta x\} = -\{f\}$.

2.3 Numerical Jacobian

code

Write a function `J = numjac(f, x, h)` that approximates the Jacobian by forward differences, one column per perturbed component of `x`. Write a verification script that tests it on the system

$$\{f\}(\{x\}) = \begin{Bmatrix} x_1^2 x_2 - 1 \\ x_1 + x_2^3 - 2 \end{Bmatrix},$$

at several values of $\{x\}$.

2.4 Multivariate Newton-Raphson

code

Write `[x, info] = newton_sys(f, J, x0, atol, maxit)` where `J` is a function handle, so that either an analytical Jacobian or `@(x) numjac(f, x, h)` can be passed in. Use one of your programs from lab 1 for the linear solve step.

2.5 Verification

code

Write `verify_newton_sys` that solves a small system with a known solution (e.g. the intersection of a circle and a line) and checks the result.

2.6 Solve for initial shock angles

writeup code

Use your code from week 1 and Equations 2 and 3 to find θ_A, θ_B, M_2, M_3, $\frac{p_2}{p_1}$ and $\frac{p_3}{p_1}$. Report the values that you found in the writeup.

2.7 Shock intersection

writeup code

Solve Eqs. (4)–(5) with `newton_sys`. Report ϕ, θ_C, θ_D, and p_4/p_1.

2.8 Initial guess sensitivity

writeup code

Repeat Task 2.7 from at least 3 initial guesses and report where Newton-Raphson converges, or that it does not converge, in each case.

Compiled 2026-09-18 04:15:42