Lab Report 2: Shock Wave Analysis

Nathaniel L. Holmes
North Carolina State University, Raleigh, NC, 27607

Nomenclature

N	=	block number
θ	=	wedge angle
β	=	shock angle
γ	=	specific heat of fluid
M	=	Mach number

I. Introduction

THIS document is a report on the second MAE 352 lab in the Spring 2019 semester. Pictures were taken of shockwaves using the Schlieren system, and the aim of the lab was to analyze the shockwave images to calculate the shock angle. The block number and wedge angle were recorded for each run, during which twenty photos were taken. Block number and wedge angle values were varied between tests to provide a set of data to be compared to the theoretical data set. MATLAB was used to process all images and create the plots.

II. Calculations

A. Mach Number by Block Number

The isentropic calibration equation from lab 1 was used to calculate the Mach number from the recorded block number. This relationship is shown below.

$$M = 1.82 * 10^{-7} * N^2 - 1.3 * 10^{-3} * N + 3.90 \quad (1)$$

B. Mach Number by Theta-Beta-Mach Relation

A solver in MATLAB was used to compute Mach number through using the equation below, the best found shock angle values, and the recorded wedge angle values.

$$\cot(\theta) = \tan(\beta) * \left(\frac{(\gamma + 1) * M^2}{2 * (M^2 \sin^2 \beta - 1)} - 1 \right) \quad (2)$$

III. Data and Results

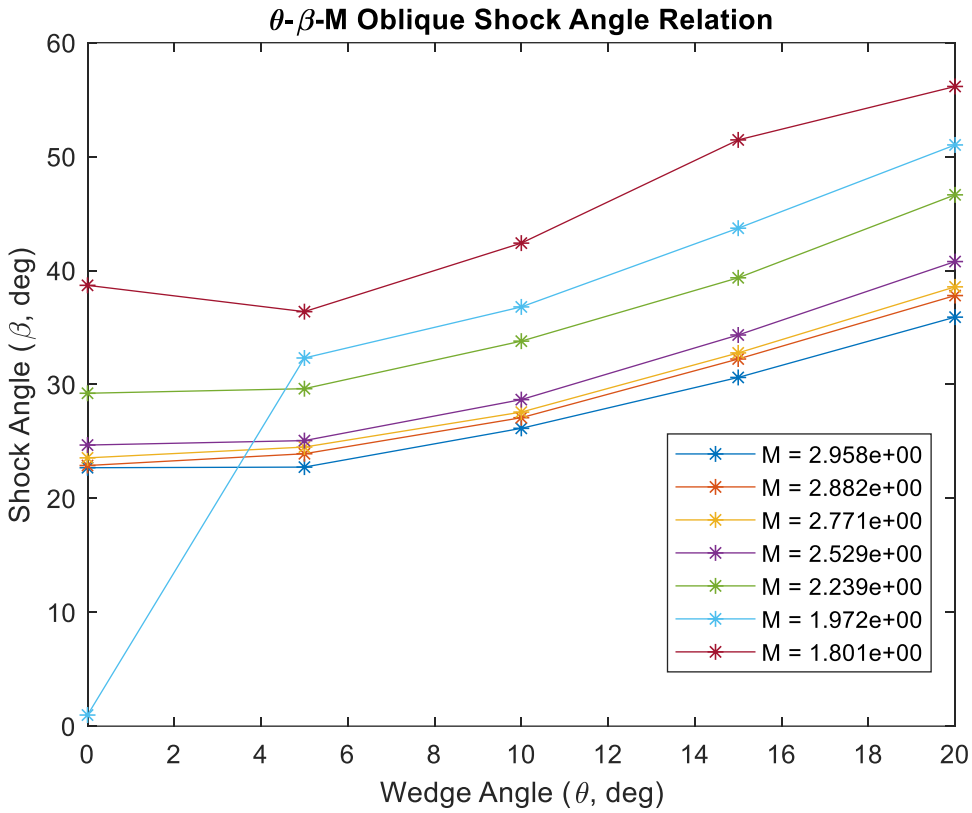

Figure 1. Theta-Beta-Mach Relation. *Seven Mach numbers are shown above, each calculated using the isentropic calibration equation previously used in Lab 1. The data points show a good overall shape with respect to the theoretical graph, except for Mach number equaling 1.972, which above has a data point that drops low at a wedge angle of zero. The shock angle value for each was based off the tenth image of each picture set.*

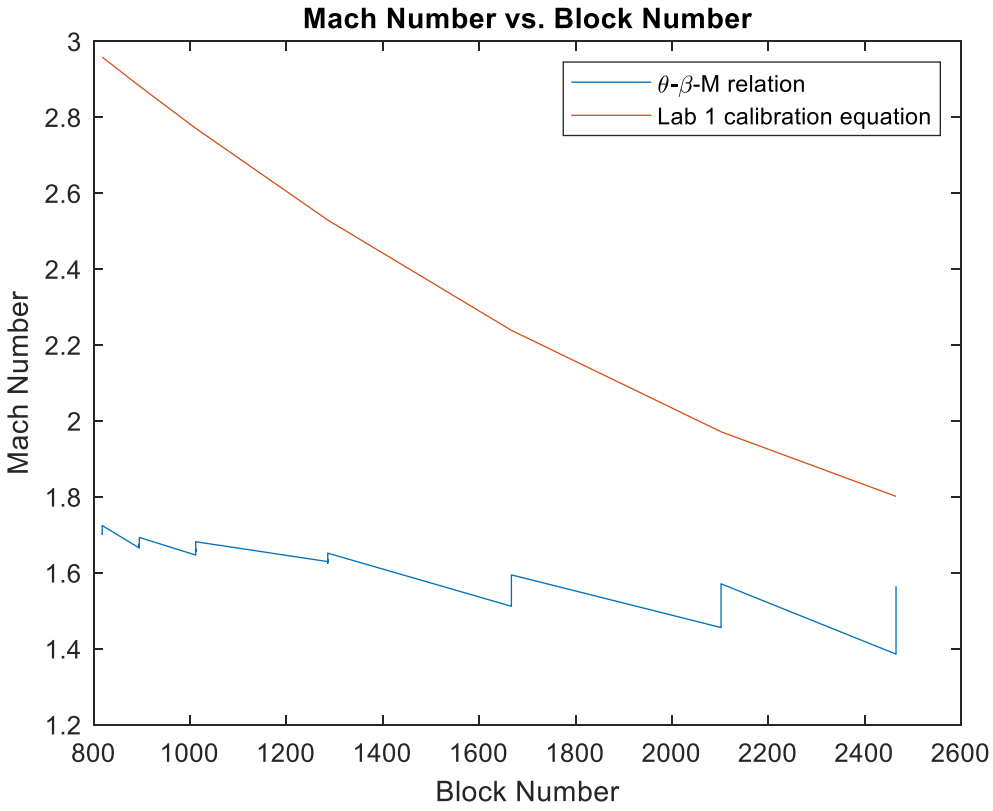

Figure 2. Mach Number vs. Block Number. *Shown above is the relationship of Mach number and block number. The theta-beta-M relation has discontinuities due to the MATLAB solver, and overall has a lower Mach number distribution than illustrated by the calibration equation from lab 1.*

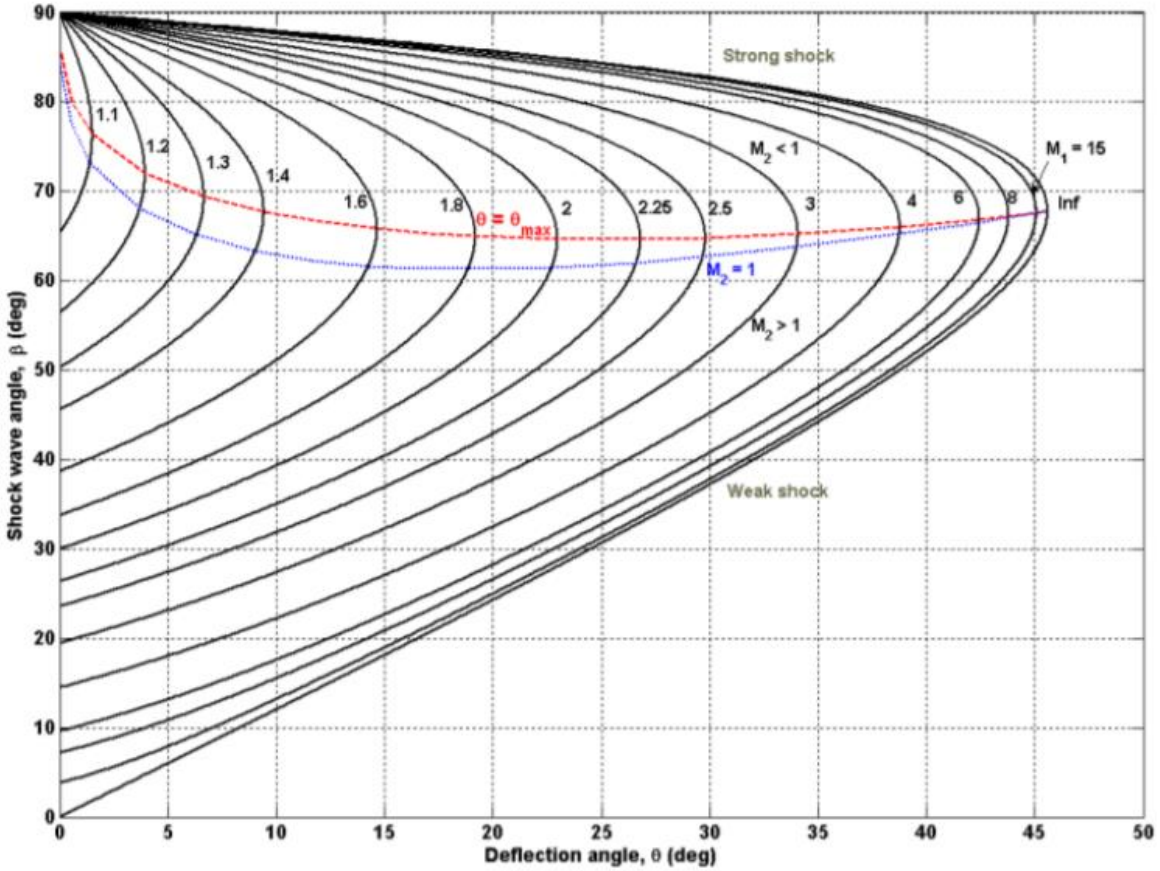

Figure 3. Theoretical Theta-Beta-Mach. *Above is the theoretical relationship image from the lab handout for reference and comparison to Fig. 1. Figure 1 represents only the lower portion of this graph, but compares well to the theoretical shape shape.*

IV. Discussion

From inspection, each set of twenty photos that corresponded to a block number and wedge angle did not have a shockwave present in the first several images and the last several images. To account for this, the first and last six images were excluded from the calculation of the shock angle value. The edge detection function often ran into issues with white clusters and incomplete shockwave edges. A threshold value named "trim_value" in the code was used to assist in the detection of these shock lines, this value was set to 0.13 to increase edge detection sensitivity. In the event that one of the shock lines was incomplete, the code was designed to ignore all pixels that were inconsistent with the shock line, and only use points close to each other during edge detection for the linear fit lines that were later calculated. Each image was also cropped on the top by one hundred pixels as to reduce the amount of processing needed. The limit was set to one hundred to provide enough room for the lower Mach number shocks to expand outward, while still truncating the size of the image to a more easily-handled size.

As seen when comparing Fig. 1 to Fig. 3, the experimental data matches up nicely with the lower quadrants of the theoretical data. There is an exception of the one data point for Mach number 1.972, however. The figure shows that as higher Mach numbers produce lower shock angle values, and as the flow moves faster, this logically makes sense. In Fig. 2, the lab one calibration equation again proved its validity, but the theta-beta-Mach relation provided lower Mach numbers than expected and shows discontinuities due to restrictions in the MATLAB solver. It could be an error in the handling of the solver that caused this problem, but the calculated values show a significant amount of deviation regardless.

V. Appendix

```
%% Aero II Lab, Lab 2: Image processing for Mach number
%% Load all files
% Get file name as string
% Get block number and store as array of values (for mach number
calculation)
% Also get theta number and do the same
% Read each image
numFolders = 35;
numPhotos = 20;

% From Lab 2 Psuedocode and then edited
% Find all files in wdir matching the pattern DSC_*.jpg
blockNumber = [];
theta = [];
currDir = dir; % gets current directory info
for k = 1:numFolders % number of folders in the data set, skips
the '.' and '..' folders
    % Get file name
    folderName{k} = currDir(k+2).name; % gets folder names and
makes cell array
                                       % k+2 for the '.' and
'..' folders again
    % Get block number and theta values
    blockNumber{k} = folderName{k}(6:9); % location of block
number in folder name strings
    theta{k} = folderName{k}(16:end);% location of theta angle
in folder name strings
    images = dir(sprintf('%s/DSC_*.jpg', folderName{k}));


    % Load all images
    for j = 1 : numPhotos % number of pictures per set
        % file name changes for each picture
        fileName = [folderName{k}, '/' , images(j).name]; %
string concatenation for file path name for imread
        pics{k}{j} = imread(fileName); % loads the current image
and stores in into pics cell array
                                       % k separates block
number
                                       % j index for picture
number
    end
end
```

```
% Convert block number and theta character cell vector values to
normal arrays of numbers
blockNumber = str2num(cell2mat(blockNumber'));
theta = str2num(cell2mat(theta'));

%% Edge detection and shock line calculations
trim_val = .13; % make like .14 for this image
for k = 1:numFolders
    for j = 1:numPhotos % number of images per set, excluding
first and last 6
        % Edge Detection
        % From Lab 2 Psuedocode
        pic = pics{k}{j}; % variable for reference to current
image
        variation1 = diff(pic);
        variation2 = diff(pic, 2);
        edgeTol1 = max(mean(variation1));
        edgeTol2 = max(mean(variation2));
        threshold = mean([edgeTol1, edgeTol2])*trim_val; %
change the .18 to some value (trim_val)

        % Perhaps an alternate method to line gaps -- skip bad
photos
        %{
        if sum(variation1 >= 10) ~= 0 %|| sum(variation2 >= 10)
~= 0
            continue % skip this photo if there is a gap in the
shock line
        end
        %}

        pic_cannys{k}{j} = edge(pic, 'canny', threshold); %
detect edges using canny method

        % Cropping
        pic_cannys{k}{j} = pic_cannys{k}{j}(100:end,:); % ignore
top 250 rows of pixels
        pic_canny = pic_cannys{k}{j}; % varaible for reference
to current edge detected image
        %figure(2)
        %imshow(pic_canny);

        % Get equation for shock lines
        % Find the uppermost white pixel in each column
        %   pic_canny is a logical matrix representing the pic,
with 1 where there is white
        [rows,cols] = size(pic_canny);
```

```
        colDex = []; % col of first white pixel index; array for
holding the indicies of the first white pixels
        rowDex = []; % row of first white pixel index; want both
rows and cols for linear fit
        count = 1; % preallocate count variable for firstDex
array
        dist = 0; % preallocate distance threshold value
        colDex(1) = 0; % preallocate
        rowDex(1) = 0; % preallocate

        %limited_range = cols/5:1:(cols - cols/5);
        for i = floor(cols/5):1:(cols - floor(cols/5)) % for
every column
            if sum(pic_canny(:,i) ~= 0) % only update array if
there's a white pixel in this col
                % get first white point values
                colTemp = i;
                rowTemp = rows - find(pic_canny(:,i), 1,
'first'); % find the first nonzero (starts at top of image)

% subtract this value from rows for plotting reasons

                % updating dist
                % only use a point that is within 10 pixels
distance of the previous point used
                if count ~= 1 % do this for all except first one
found
                    %colDex(count - 1)
                    %rowDex(count - 1)
                    dist = sqrt((colTemp - colDex(count - 1))^2
+ ...
                        (rowTemp - rowDex(count - 1))^2);
                end

                if dist <= 10 % only update arrays if the point
proximity condition is satisfied
                    colDex(count) = colTemp;
                    rowDex(count) = rowTemp;

                    count = count + 1; % update count for next
index
                end
            end
            % Shock fitting and values
            shockFit{k}{j} = polyfit(colDex,rowDex,1);
            shockVals{k}{j} = polyval(shockFit{k}{j},colDex);
        end
```

```
    end
end

% White clusters will appear away from shock depending on
trim_value (lower
%   value means more clusters will get through, cropping will
help this
%   issue)
% Cropping to predefined size
%   Find where the white shock lines are and adda buffer to top
and bottom,
%   then remove all other image material beyond the top and
bottom buffer



%imshow(pic_canny);
%% Calculate shock angle from equation
%{
%   Use equation from last lab to get the Mach number from the
block number
%   For the image cycling loop extract the block number which is
used for
%   the shock angle calculation
%theta = acot(tan(beta)*((gamma+1)*M^2/(2*(M^2*sin(beta) - 1)) -
1);
% theta is wedge angle, we want beta for shock angle

% attempt with the equation
gamma = 1.4; % specific heat ratio of air for T < 1000K
%shockAngle = 1:length(blockNumber);
for i = 1:length(blockNumber) % calculate mach number for every
block number
    M(i) = 1.82e-7 * blockNumber(i)^2 - 1.3e-3 * blockNumber(i)
+ 3.9; % Isentropic approximation equation for NCSU supson WT

% BN is block number
    syms beta
    eqn = cotd(theta(i)) ==
tand(beta)*((gamma+1)*M(i)^2/(2*(M(i)^2*sind(beta) - 1)) - 1);
    currBeta = double(vpasolve(eqn,beta));
    if isempty(currBeta) == 1
        currBeta = 0;
    end
    shockAngle(i) = currBeta;
```

```
end
%}
%% attempt with the fit line -- this seems to give more
reasonable values
%betaLine = atand(shockVals{end}{end}(end)./colDex(end));
betaLine = {};
bestPics = {};
betaVariance = {};
betaDiff = [];
%bestBeta = [];
for k = 1:numFolders % number folders (mach and theta number)
    for j = 1:numPhotos % number images per folder, but exclude
the first and last 6
        % All points will have same slope, only need to get
angle once
        %xdist = 1:length(shockVals{k}{j}(end));
        %betaLine{k} = atand(shockVals{k}{j}(end)/xdist(end)); %
will only have a beta val for each mach number

        for i = 1:length(shockVals{k}{j}) % for every point on
line
            %xdist = 1:length(shockVals{k}{j}(i)); % number of
pixel columns
            %betaLine{k}{j}(i) =
atand(shockVals{k}{j}(i)/xdist);
            betaLine{k}{j} = abs(atand(shockFit{k}{j}(1))); %
gets slope from each fit
            meanBeta = mean(cell2mat(betaLine{k})); % mean of
current beta set values
            %betaVariance{k} = diff(cell2mat(betaLine{k}));
            betaDiff = abs(cell2mat(betaLine{k}) - meanBeta); %
difference between mean and current value of beta
            minBetaDiff = min(betaDiff); % minimum difference in
beta
            %bestBeta{k} = betaLine{k}{betaDiff ==
minBetaDiff(k)};

        end
    end
    % Calculating best picture to use
    bestBeta(k) = betaLine{k}{10};
    %bestPics{k} = min(betaLine{k}{:});
end

%% Mach number calculation
tbMach = [];
for i = 1:length(blockNumber)
```

```
    M(i) = 1.82e-7 * blockNumber(i)^2 - 1.3e-3 * blockNumber(i)
+ 3.9; % Isentropic approximation equation for NCSU supson WT

% BN is block number

    % theta beta mach relation method
    gamma = 1.4;
    syms tbM
    eqn = cotd(theta(i)) ==
tand(bestBeta(i))*((gamma+1).*tbM.^2./(2.*(tbM.^2.*sind(bestBeta
(i)) - 1)) - 1);
    mSol = double(vpasolve(eqn,tbM));
    mSol = mSol(mSol > 0); % take only positive solution
    tbMach{i} = mSol;
end
M = double(M);

tbMach = tbMach(~cellfun('isempty',tbMach)); % remove empty
cells
tbMachAvg = [];
p = 1;
for i = 1:7 % number of wedge angles times block number (mach
number)
    tbMachAvg(i) = mean(cell2mat(tbMach(p:p+3))); % only have 4
mach numbers per angle now
    p = p + 4; % update index variable
end
%%
figure(1)
plot(theta(1:5),bestBeta(1:5),'*-')
hold on
plot(theta(6:10),bestBeta(6:10),'*-')
plot(theta(11:15),bestBeta(11:15),'*-')
plot(theta(16:20),bestBeta(16:20),'*-')
plot(theta(21:25),bestBeta(21:25),'*-')
plot(theta(26:30),bestBeta(26:30),'*-')
plot(theta(31:35),bestBeta(31:35),'*-')
mach = {};
for i = 1:7
    mach{i} = sprintf('M = %.3s',M(i*5));
end
title('\theta-\beta-M Oblique Shock Angle Relation')
legend(mach{1:7}); % data is block number separated (mach number
representation)
xlabel('Wedge Angle (\theta, deg)');
ylabel('Shock Angle (\beta, deg)');
%{
```

```
%for f = 1:7 % number of unique block numbers
   while q <= 31 % number of unique wedge angles
       plot(theta(q:q+4), bestBeta(q,q+4), '*-');
       hold on
       q = q + 1;
   end
   hold off
%end
%}
%plot(theta,-shockAngle)
hold off
%% plot mach number vs block number
tbBlockNumber = [];
for i = 1:5:35
    tbBlockNumber = [tbBlockNumber; blockNumber(i+1:i+4)];
end
figure(2)
plot(tbBlockNumber,cell2mat(tbMach)) % discontinuities due to
solver clearly seen, mach lower
hold on
plot(blockNumber,M);
xlabel('Block Number');
ylabel('Mach Number');
title('Mach Number vs. Block Number');
legend('\theta-\beta-M relation','Lab 1 calibration equation');
```

VI. References

[1]Narsipur, Shreyas. "MAE 352 – Experimental Aerodynamics II Lab 2 – Shock Wave Analysis". *NCSU,* February 5, 2019.