Report 6: Oblique Shock Wave

Submitted to:

Prof. First Last

By:

|  |  |  |
| --- | --- | --- |
|  | احمد محمد محمود | Sec. 1, B.N. 30 |
|  | Name Name | Sec. 1, B.N. 31 |
|  | Name Name | Sec. 2, B.N. 32 |

*This page intentionally left blank!*

# Contents

[Contents i](#_Toc482719673)

[Nomenclature i](#_Toc482719674)

[1. Problem Statement 1](#_Toc482719675)

[2. Mathematical Model 1](#_Toc482719676)

[3. Assumptions 1](#_Toc482719677)

[4. Analysis 1](#_Toc482719678)

[4.1. Working Procedure: 2](#_Toc482719679)

[4.2. Results 3](#_Toc482719680)

[5. Conclusion 3](#_Toc482719681)

[Appendices 5](#_Toc482719682)

[A. Matlab Codes 5](#_Toc482719683)

[References 7](#_Toc482719684)

# Nomenclature

N.S. Navier Stockes

w.r.t. with respect to

*This page intentionally left blank!*

# Problem Statement

Produce charts that describe the change of supersonic flow properties when it turned away from itself.

# Mathematical Model

The most general governing equation is N.S. equation. This is any dummy text just to show the capabilities of nomenclatures *θ* of LyX w.r.t. LaTeX.

These equations are bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

# Assumptions

1. Steady flow
2. Quasi-dimensional flow; (area is variable with x only).
3. Bla bla bla bla bla bla bla bla bla bla bla bla.
4. Bla bla bla bla bla bla bla bla bla bla bla bla.
5. Bla bla bla bla bla bla bla bla bla bla bla bla.
6. Bla bla bla bla bla bla bla bla bla bla bla bla.
7. Bla bla bla bla bla bla bla bla bla bla bla bla.
8. Body forces can de neglected; (weight of fluid)
9. Viscous stresses are absent
10. Changes in potential energy are neglected
11. Perfect gas
12. Thermally perfect gas
13. Adiabatic flow with no external work

# Analysis



We can derive the formula that governs super flow expansion as:

## Working Procedure:

If you know & know you can use equation (3) to solve for using Newton Raphson iteration scheme as described below:

After you get you can get the pressure and temperature using the isentropic relations:

**F**unction M\_2o.m [Code 2] illustrates this procedure. Report6 main.m [Code 1] was used to plot the figures shown in section ‎4.2. Report6 main.m [Code 3] is just for exporting the figure to a pdf or emf to be included in section ‎4.2. Therefore this code is not complete.

This is a line with some Arabic words هذا هو بعض الكلمات العربية فى سطر انجليزى.

و هذا سطر عربى به بعض الكلمات الانجليزية.Thus is some English words in an Arabic line.

و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب.  و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب. و بناءً عليه فقد تم اثبات المطلوب.

## Results



The following table is just for demonstration. It doesn’t provide any useful information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Angle (*θ*)** | **Temperature** | **Pressure** | **Density** |
| 12 | 324 | 6780 | 7676 |
| 12 | 232 | 232 | 7565 |
| 23 | 12121 | 232 | 4654 |

# Conclusion

We see that we can still obtain solutions for for . But, however I think the solutions for aren’t practical.

Pressure, temperature & density increases as the kinetic energy increase as in ‎[1].

*This page intentionally left blank!*

# Appendices

# Matlab Codes

Code : Report6 main.m

|  |
| --- |
| function M\_2\_vec=M\_2o(M\_1,theta\_d\_vec, ...  gamma) %optional arguments  if nargin<3  gamma=1.4;  end    theta\_vec=deg2rad(theta\_d\_vec);  n1=sqrt((gamma+1)/(gamma-1))\*atan(sqrt((gamma-1)/(gamma+1)\*(M\_1^2-1)))-atan(sqrt(M\_1^2-1));    %Newton Raphson iteration  M\_2\_vec=1.1\*ones(size(theta\_d\_vec));  for ii=1:length(theta\_d\_vec)  f=sqrt((gamma+1)/(gamma-1))\*atan(sqrt((gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1)))-atan(sqrt(M\_2\_vec(ii)^2-1))-theta\_vec(ii)-n1;  fdash=1/(M\_2\_vec(ii)^2-1)^(1/2)\*M\_2\_vec(ii)/(1+(gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1))-1/(M\_2\_vec(ii)^2-1)^(1/2)/M\_2\_vec(ii);    M\_2\_n=M\_2\_vec(ii)-f/fdash;  while abs(M\_2\_vec(ii)-M\_2\_n)>=100\*eps %This is dangerous. Infinte loop can occur!!  M\_2\_vec(ii)=M\_2\_n;  f=sqrt((gamma+1)/(gamma-1))\*atan(sqrt((gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1)))-atan(sqrt(M\_2\_vec(ii)^2-1))-theta\_vec(ii)-n1;  fdash=1/(M\_2\_vec(ii)^2-1)^(1/2)\*M\_2\_vec(ii)/(1+(gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1))-1/(M\_2\_vec(ii)^2-1)^(1/2)/M\_2\_vec(ii);  M\_2\_n=M\_2\_vec(ii)-f./fdash;  end  M\_2\_vec(ii)=M\_2\_n;  end |

Code : Function M\_2o.m

|  |
| --- |
| function M\_2\_vec=M\_2o(M\_1,theta\_d\_vec, ...  gamma) %optional arguments  if nargin<3  gamma=1.4;  end    theta\_vec=deg2rad(theta\_d\_vec);  n1=sqrt((gamma+1)/(gamma-1))\*atan(sqrt((gamma-1)/(gamma+1)\*(M\_1^2-1)))-atan(sqrt(M\_1^2-1));    %Newton Raphson iteration  M\_2\_vec=1.1\*ones(size(theta\_d\_vec));  for ii=1:length(theta\_d\_vec)  f=sqrt((gamma+1)/(gamma-1))\*atan(sqrt((gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1)))-atan(sqrt(M\_2\_vec(ii)^2-1))-theta\_vec(ii)-n1;  fdash=1/(M\_2\_vec(ii)^2-1)^(1/2)\*M\_2\_vec(ii)/(1+(gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1))-1/(M\_2\_vec(ii)^2-1)^(1/2)/M\_2\_vec(ii);    M\_2\_n=M\_2\_vec(ii)-f/fdash;  while abs(M\_2\_vec(ii)-M\_2\_n)>=100\*eps %This is dangerous. Infinte loop can occur!!  M\_2\_vec(ii)=M\_2\_n;  f=sqrt((gamma+1)/(gamma-1))\*atan(sqrt((gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1)))-atan(sqrt(M\_2\_vec(ii)^2-1))-theta\_vec(ii)-n1;  fdash=1/(M\_2\_vec(ii)^2-1)^(1/2)\*M\_2\_vec(ii)/(1+(gamma-1)/(gamma+1)\*(M\_2\_vec(ii)^2-1))-1/(M\_2\_vec(ii)^2-1)^(1/2)/M\_2\_vec(ii);  M\_2\_n=M\_2\_vec(ii)-f./fdash;  end  M\_2\_vec(ii)=M\_2\_n;  end |

Code : Function export\_figure

|  |
| --- |
| unction export\_figure(h\_vec, ...  Expand,filenames,resolution,pictureFormat) %Optional arguments    if nargin<2  Expand='';  end    if nargin<4  resolution=600;  elseif isempty(resolution)  resolution=600;  end    if nargin<5  pictureFormat={'pdf'};  else  if ~iscell(pictureFormat)  error('pictureFormat must be cell array of strings.')  end  end    %%% A lot more code |

# 

# References

1. J. D. Anderson, Modern Compressible Flow, McGraw-Hill, New York, 1990.
2. Report (1).
3. Report (3).