
Supersonic Nozzle Thrust Optimization Using MATLAB and CFD

Rezvan Abdollahi

Contents

Introduction	4
Initial Test case	4
Basic geometry.....	4
Flow parameter and CFD Solver	5
Results	8
Optimization process	10
Geometry Parameterization.....	10
Artificial Network Building.....	11
Optimization Loop.....	13

Final Algorithm	15
Results and discussion.....	16
5 control points	16
4 points with constraints on throat area.....	18
Final results	21
Conclusion.....	24
Appendix.....	25
Main code	25
Control point and profile generation.....	26
GAMBIT AND Fluent Run	28
Thrust Calculator.....	29
data base for neural net works	30
pressure and velocity thrust networks.....	31
Pressure and velocity thrust calculator.....	32
Updating network database	34
Thrust ratio Constraints	35

Figures

Figure 1:Initial Mesh.....	5
Figure 2:Type of material and its governing equation	5
Figure 3: Solver	6
Figure 4:Turbulence Model.....	6
Figure 5: Inlet Boundary Condition	7
Figure 6: Outlet Boundary Condition	7
Figure 7: Convergence Criteria.....	8
Figure 8: Mach Contour	8
Figure 9:Pressure Contour	9
Figure 10: Nozzles generated via a 4 control points approach.....	11
Figure 11:Thrust due to pressure difference network	12
Figure 12:Thrust due to velocity difference network	13
Figure 13:Inputs of neural networks.....	16
Figure 14:Final outputs after 60 iterations.....	17
Figure 15:Mesh-5 points	17
Figure 16:Mach contour- 5points	18
Figure 17:Grid for forced throat.....	19
Figure 18: Mach contour for forced throat.....	19
Figure 19:Grid for not forced throat.....	20
Figure 20: Mach contour for not forced throat	20
Figure 21:Final converged profiles with highest out put thrust.....	21
Figure 22:Final answer grid.....	22
Figure 23:Final answer Mach contour.....	22
Figure 24:Final answer pressure contour.....	23
Figure 25:Final answer pressure and velocity convergence history	23

Tables

Table 1:Upper and lower bands of inputs	13
Table 2: Constraints -4 points approach.....	14
Table 3: Constraints -5 points approach.....	14

Introduction

The aim this report is to explain an optimization process for maximizing the thrust production for a supersonic nozzle. The optimization algorithm is a combination of CFD analysis, artificial neural network training and a constrained function minimization. At the beginning we will discuss the initial test case characteristics and the goal of optimization, then the optimization algorithm will be discussed. Finally, there would be the results and discussion about different issue during the whole process.

Initial Test case

At the beginning of the process, a test case has been solved via GAMBIT-FLUENT solver. The purpose of study is maximizing thrust production of convergence divergence nozzle. Since the converging part is assumed to remain unchanged during the process, the diverging supersonic part is subject of optimization only. The main motivation of solving this test case is production of the main journals. While we need to produce database for the neural network training and running GAMBIT-FLUENT during the optimization cycle, the accurate production of the journal files is crucially important. The major step of building the journal files will be discussed as follows

Basic geometry

The basic initial geometry is the divergence part of a CD-nozzle. The throat and the inlet flow characteristics has been set in a way that the flow enters with a Mach number close to one, however while the geometry may alter during the process this condition will not remain the same for all the test cases.

A sample mesh that has been generated using the main GAMBIT journal has been shown in the following figure.

It should be mention that during the entire process the length of the nozzle and the diameter of inlet flow will remain constant. The length of nozzle is set to be 0.85 meters and the inlet diameter is 0.2636 m. In order to decrease the runtime, the nozzle assumed to be axisymmetric therefore the mesh has been only generated for the half upper part. The mid-line of the nozzle is set to satisfy the symmetry boundary condition.

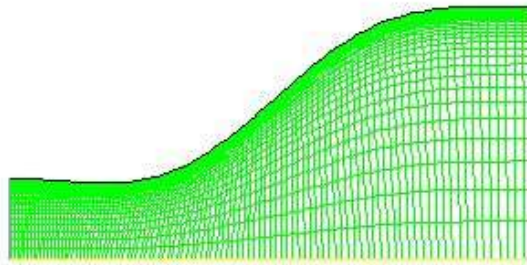


Figure 1:Initial Mesh

Flow parameter and CFD Solver

Input parameter including, the inlet and out let boundary condition, type of the solver and turbulence model, convergence criteria and the target outputs have been described in the following figures

The working fluid is assumed to air and the governing equation is ideal gas

Figure 2:Type of material and its governing equation

Since the flow regime is compressible, the solver will be density base and the axisymmetric condition will be applied here as well

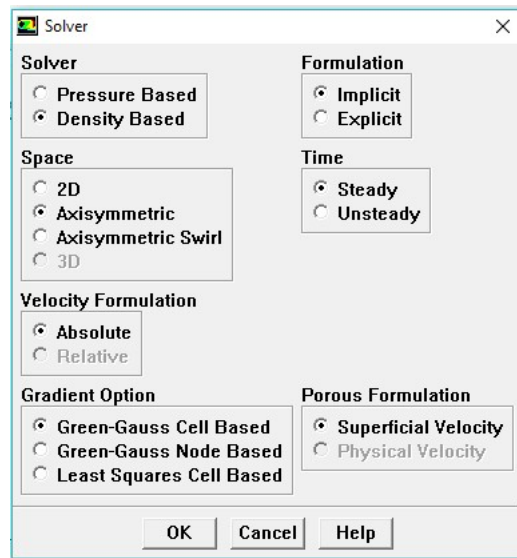


Figure 3: Solver

Based on a brief study in other similar problem the turbulence model is selected to be $k-\omega$ sst.

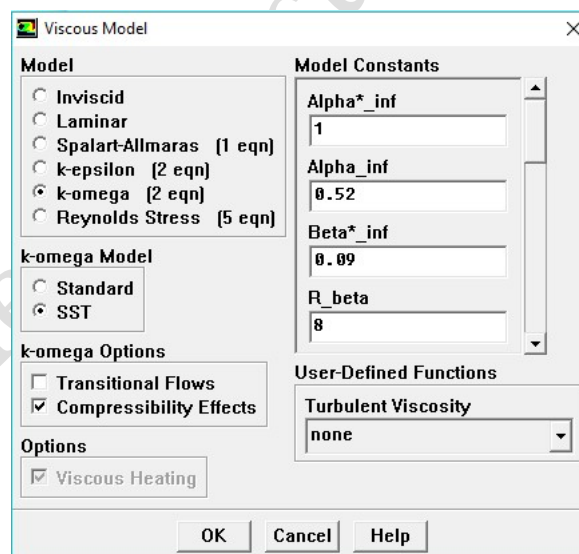
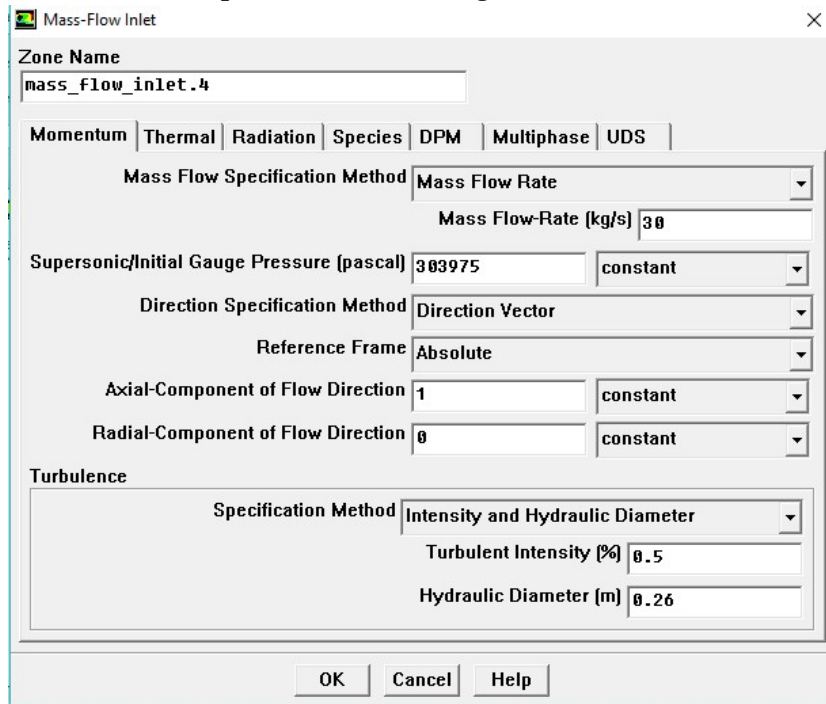


Figure 4: Turbulence Model

The static pressure is almost 4 bar and the mass flow inlet is 30 kg/s

Outlet surface boundary condition is pressure outlet and the convergence criteria for each flow parameter is 10^{-5} . Since the accuracy of the CFD results does not major concern here, the number of the iteration is limited to be 5000 whether the

problem is converged or not.



Mass-Flow Inlet

Zone Name
mass_flow_inlet.4

Momentum | Thermal | Radiation | Species | DPM | Multiphase | UDS

Mass Flow Specification Method: Mass Flow Rate
Mass Flow-Rate (kg/s): 30

Supersonic/Initial Gauge Pressure (pascal): 303975 constant

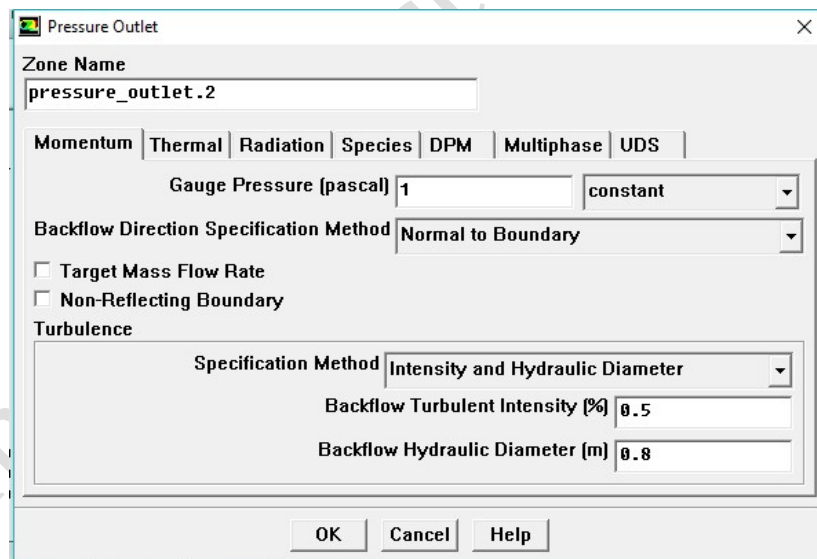
Direction Specification Method: Direction Vector
Reference Frame: Absolute

Axial-Component of Flow Direction: 1 constant
Radial-Component of Flow Direction: 0 constant

Turbulence
Specification Method: Intensity and Hydraulic Diameter
Turbulent Intensity (%): 0.5
Hydraulic Diameter (m): 0.26

OK Cancel Help

Figure 5: Inlet Boundary Condition



Pressure Outlet

Zone Name
pressure_outlet.2

Momentum | Thermal | Radiation | Species | DPM | Multiphase | UDS

Gauge Pressure (pascal): 1 constant

Backflow Direction Specification Method: Normal to Boundary

☐ Target Mass Flow Rate
☐ Non-Reflecting Boundary

Turbulence
Specification Method: Intensity and Hydraulic Diameter
Backflow Turbulent Intensity (%): 0.5
Backflow Hydraulic Diameter (m): 0.8

OK Cancel Help

Figure 6: Outlet Boundary Condition

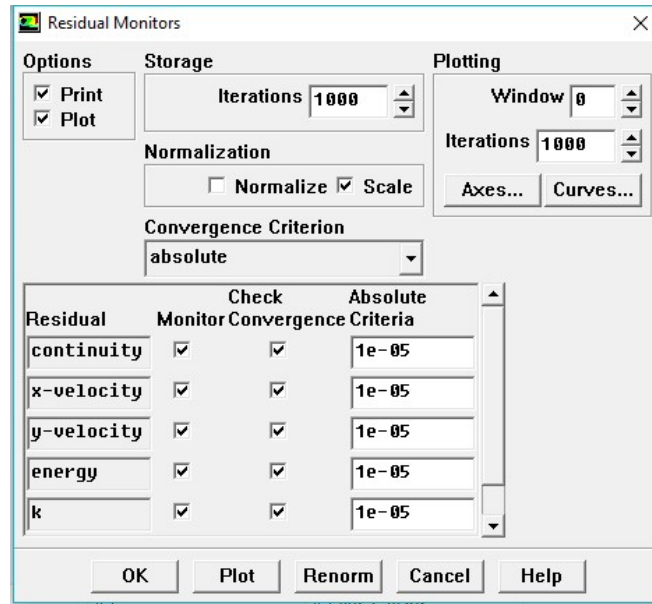


Figure 7: Convergence Criteria

Results

Based on above CFD analysis condition, the results for the initial test case are shown in following figures.

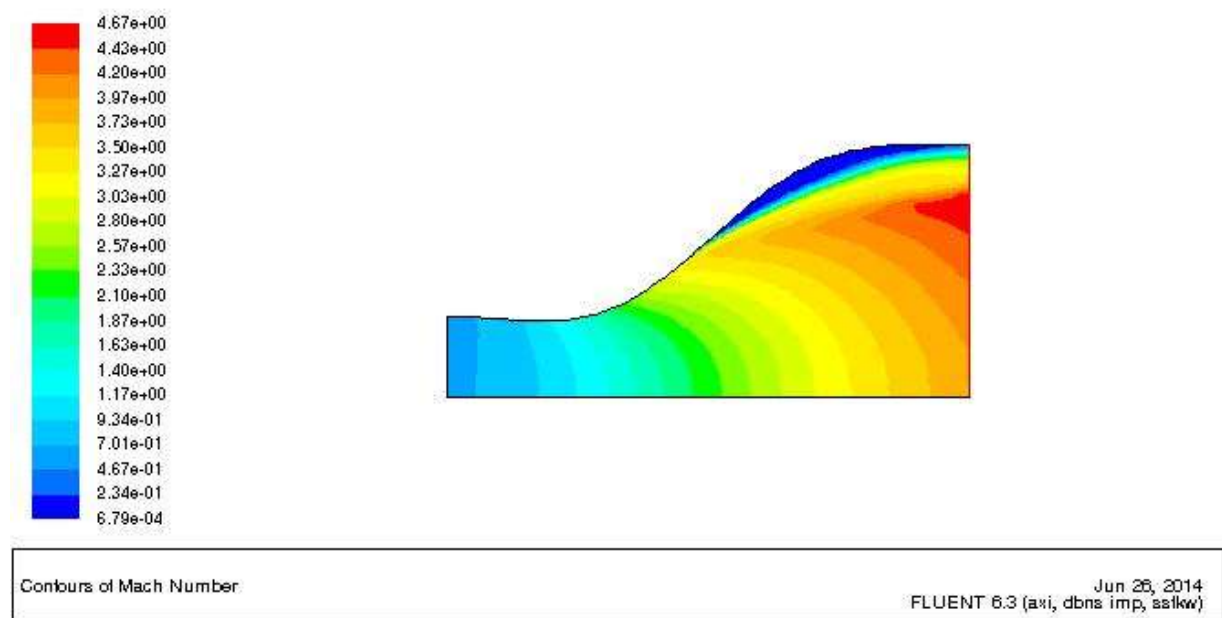


Figure 8: Mach Contour

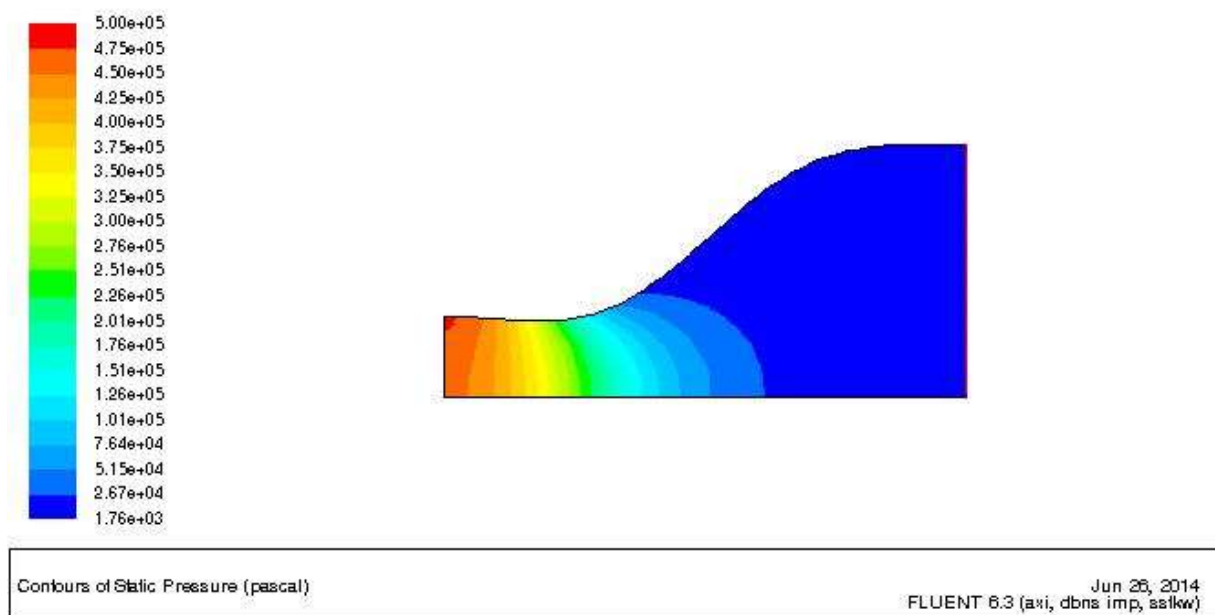


Figure 9:Pressure Contour

For this test case the inlet Mach No. is 0.631 and the outlet Mach No. is 1.833. for this specific test case thrust due to velocity is 9.6204×10^3 N and due to pressure difference is - 9.5281×10^3 N.

Optimization process

For solving the problem, we have to propose a model for geometry generation, an objective function to be utilized in the optimization and a optimization method to achieve the final results. The major steps are described in upcoming sections.

Geometry Parameterization

For achieving a generalized geometry builder function, two main MATLAB function have been developed. First, we have to generate the control points which can create an arbitrary nozzle shape. These points are actually the inputs of the final objective function. For control point generation, *Cont_pt_generator* has been utilized, this function produces random control points with following constraints:

1. The base of point production is predefined; the points have been selected in a way that can produce a diverging nozzle even if there would be a slight change in their y coordinates.
2. the next generation of point has to change randomly with a limited deviation relative to the initial points,
3. The trend of each points set is forced to be increasing
4. The first point is forced to always be the same

In the point generation process following assumption has been made

1. The x-coordinates of points assumed to be constant and only the y coordinate will change
2. The deviation of each point in every generation is proportional to its initial value.
3. The value of deviation regarding their initial coordinates has been set to be 0.1 at first, while the final shapes did seem to cover the optimum thrust nozzle, the deviation has been replaced by 0.15

The next step is to produce the inputs file for the GAMBIT journal. In this step the points have been extracted from a spline passing through the generated control points. The spline has to have zero slop at the beginning and the end of itself. Finally, 16 point will be extracted from the spline and saved as GAMBIT journal inputs file. MATLAB function *point_gen* is in charge of the above process.

Some samples of generated nozzle shapes have been shown in the following figure.

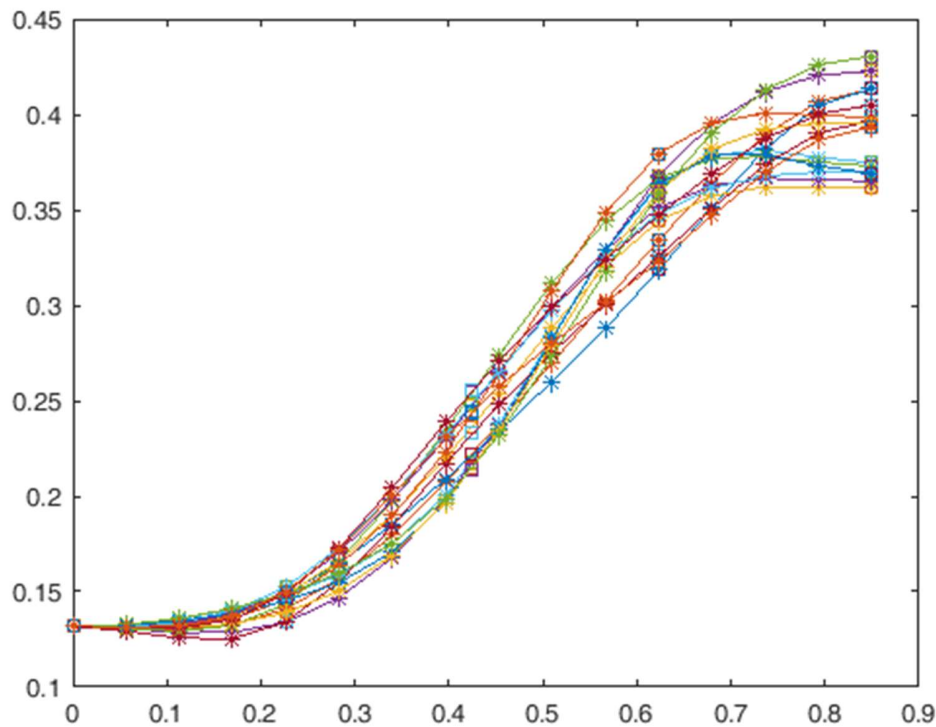


Figure 10: Nozzles generated via a 4 control points approach

Artificial Network Building

For solving the problem, we do need to develop an objective function, therefore, we will use the artificial neural network. Building the objective function have the following steps

1. Running GAMBIT-FLUENT for finite numbers of randomly generated geometries, using *gambit_run.m* and *fluent_run.m*
2. Reading the pressure and velocity data in the inlet and out let of each test case *thrust_clc.m*
3. Production of the thrust function which is a combination of pressure and velocity related parts, using Creating a neural network with inputs of control points y coordinate and pressure and velocity thrust. using *ann_builder_tv.m* and *ann_builder_tp.m*
4. Saving the networks to be used as an objective function that returns thrust vector for any arbitrary inputs, using *thrust_opt.m*

In building the neural network there were some considerations

- There was a study to obtain the best number hidden layer, nodes and enough inputs data, finally the hidden layer set to be 3 with 5 neuron

in each layer and the number of initial set for training the network is assumed to be 15

- At first the network has been built in a way that with one input vectors of y coordinates returned a couple of data for pressure and velocity related thrust, however, the final results from optimization demonstrated some in consistency and the convergence was hardly achieved the MATLAB code responsible for this part was *ann_builder.m*. Therefore, the new strategy was to divide the thrust data and train two separated networks for each part. The functions are described above
- The whole process of training the networks for the first be carried by *outout_loop.m*

The results of training each network can be seen in the following figures. These data have been extracted after the final loop of training for each network. The number of the inputs for each one is more than 100 samples.

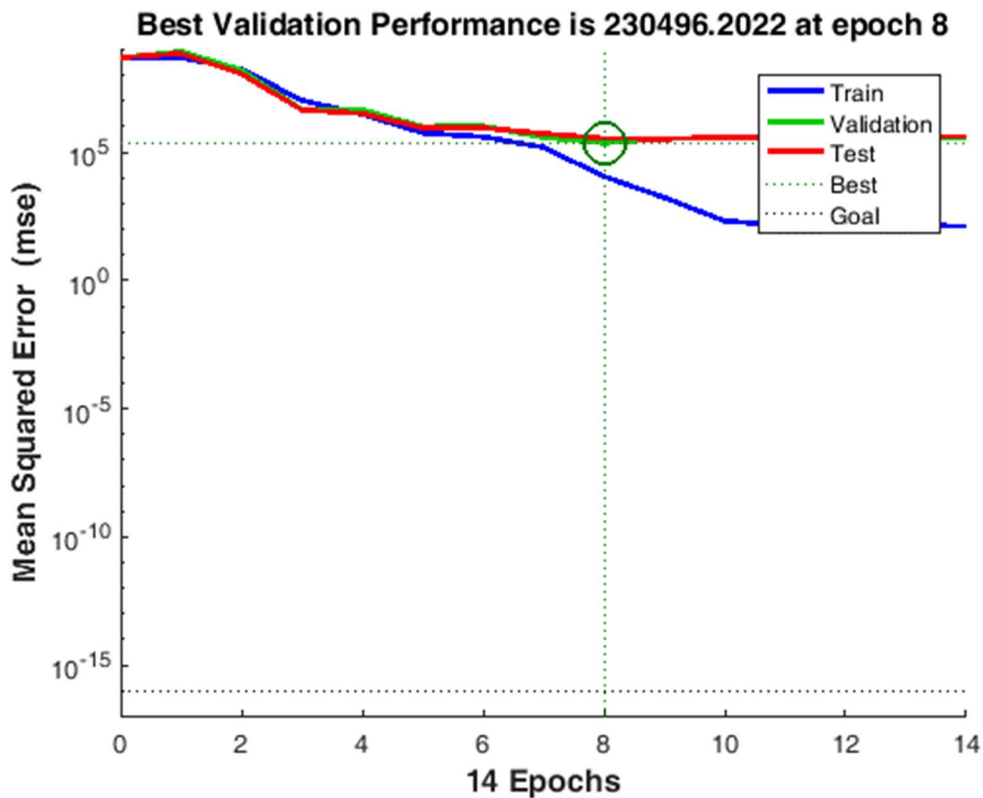


Figure 11: Thrust due to pressure difference network

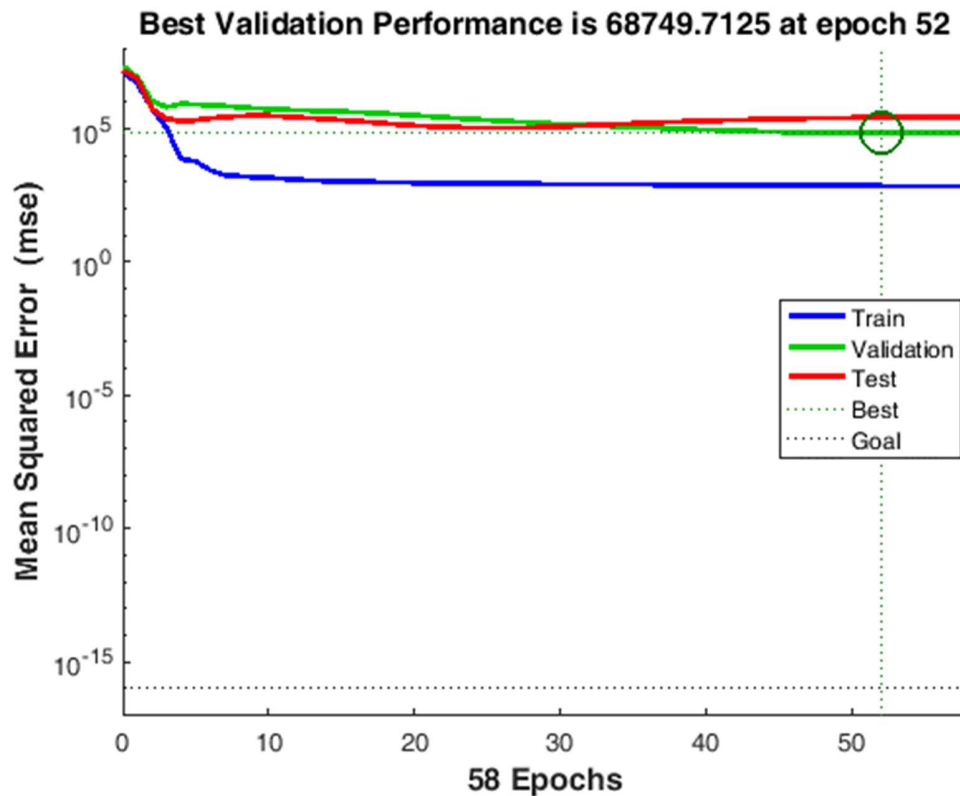


Figure 12: Thrust due to velocity difference network

Optimization Loop

Optimization is an iterative process that has following steps

1. Reading initial values for y coordinate
2. Setting the upper and the lower constrained for all variable, these range have been shown in the following table, the trend has been chosen in a way that the convergence will be obtained with minimum iterations

Table 1: Upper and lower bands of inputs

Variable	Upper band	Lower band
Y(1)	Y(1)	Y(1)
Y(2)	Y(2)+20%Y(2)	Y(2)-10%Y(2)
Y(3)	Y(3)+10%Y(3)	Y(3)-10%Y(3)
Y(4)	Y(4)+05%Y(4)	Y(3)-05%Y(4)

3. Setting the problem constraints, since the problem has been solved via a 5 control points and 4 control points approach there are two different set of constraints. It should be mentioned here that all the constraints in this problem are linear

Table 2: Constraints -4 points approach

4 points	
Constrain	Equation
Fixing first point	$Y(1)=0.1318$
Increasing trend	$Y(1)-Y(2)<0$
Increasing trend	$Y(2)-Y(3)<0$
Increasing trend	$Y(3)-Y(4)<0$

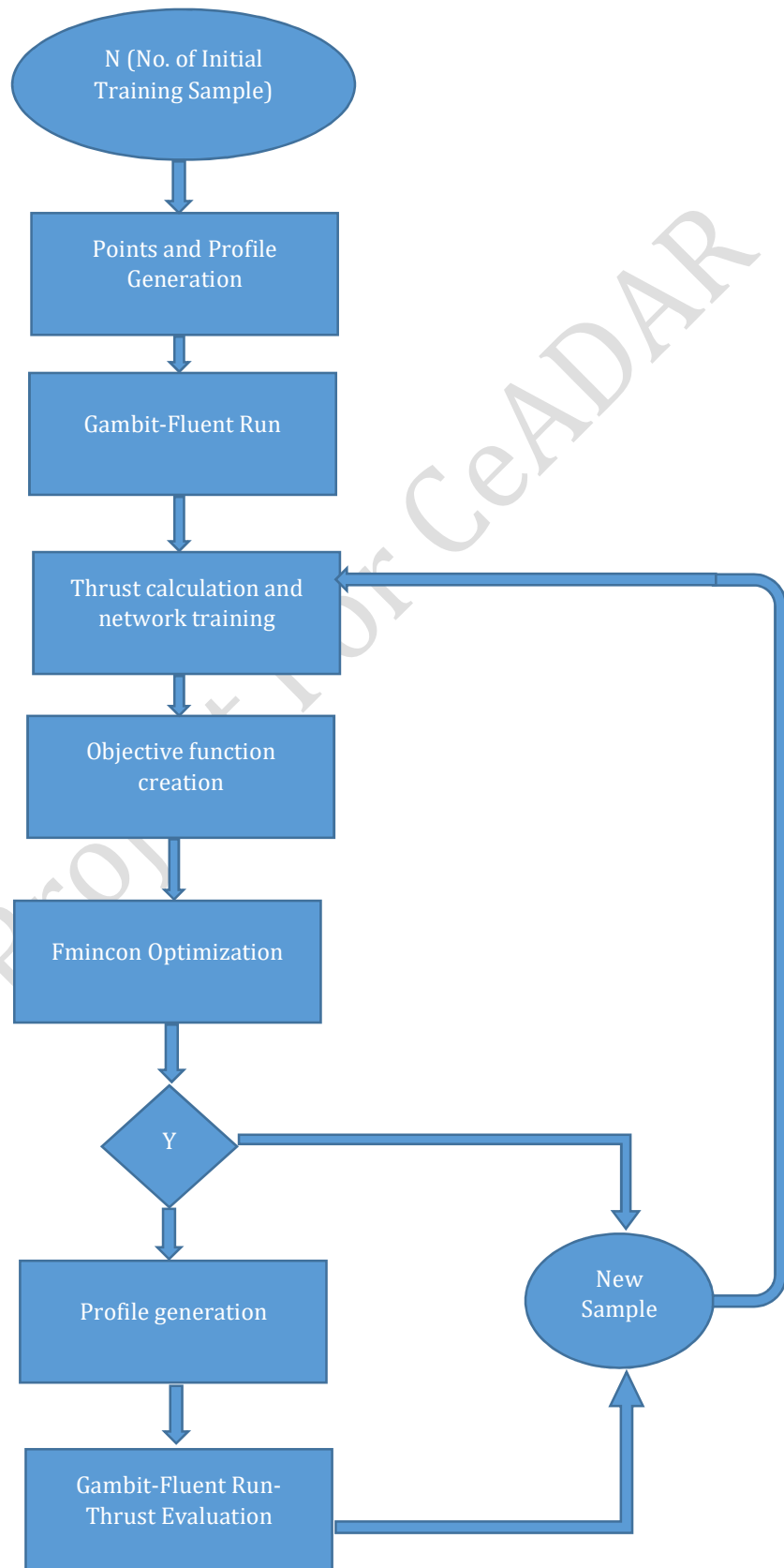
4. Hence, in the 4-point system the constraint for zero slop at the end will be satisfied automatically in 90 percent of test cases. therefore, there is no need to apply it separately. However, in the 5-point system this should added to the constraints. The corresponding equations have been extracted from central difference, 2nd order forward difference and 2nd order backward difference respectively

Table 3: Constraints -5 points approach

5 points	
Constrain	Equation
Fixing first point	$Y(1)=0.1318$
Increasing trend	$Y(1)-Y(2)<0$
Increasing trend	$Y(2)-Y(3)<0$
Increasing trend	$Y(3)-Y(4)<0$
Increasing trend	$Y(4)-Y(5)<0$
Positive 2 nd derivative at point 2	$-Y(1)+2Y(2)-Y(3)<0$
Negative 2 nd derivative at point 4	$Y(3)-2Y(4)+Y(5)<0$
Zero slope at the inlet	$3Y(1)-4Y(2)+Y(3)=0$
Zero slope at the outlet	$-3Y(3)+4Y(4)-Y(5)=0$

5. Starting the optimization process via *fmincon* and with *@Thrust_opt* as the objective function as a result the Y will be extracted from the process
6. Generating the inputs data for GAMBIT journal using the Y as control points
7. Running the GAMBIT and FLUENT for the generated points and obtain the results
8. Adding the result for the new point to the neural networks database via *data_recorder.m*
9. Re training the networks
10. Start the optimization again till the output profile remains the same in three in row iterations

Final Algorithm



Results and discussion

Obtaining a feasible result was a very hard goal to achieve! The different types of parameters including flow, geometry and optimization parameters have been tested and various strategies have been applied to achieve the final goal. First the results for some of these efforts will be discussed and then the final feasible data will be illustrated. The difficulty in the process was due to non-linearity of the problem, since the existence of shock waves and altering the inlet condition during the optimization process made it difficult for the algorithm to satisfy all the constraints and converge to a feasible point.

5 control points

In this section results for the following condition have been described.

- 5 control point
- condition on first and 2nd derivatives
- 1 trained network for both thrusts
- Inlet mass flow rate 20 kg/s
- With and without force on throat area

Even though the inputs profile for each test cases satisfied the constraints the final converged results are in-feasible the inputs and outputs after 60 iterations are shown in following figures.

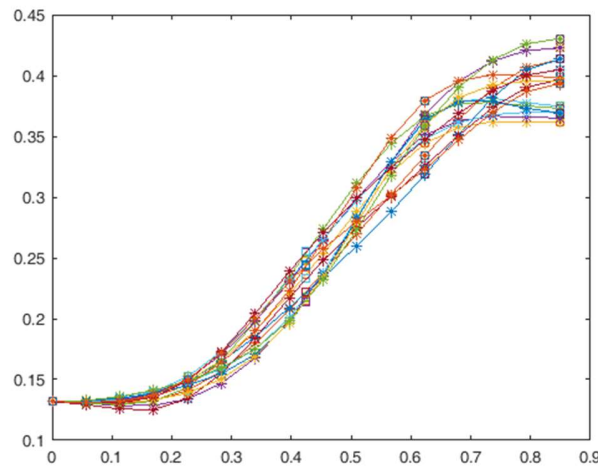


Figure 13: Inputs of neural networks

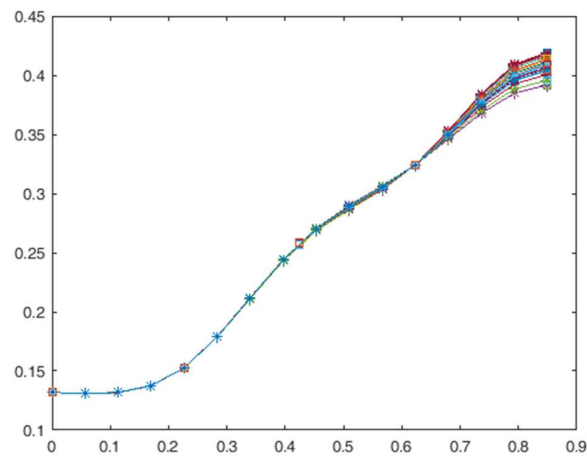


Figure 14: Final outputs after 60 iterations

The CFD results for the final converged case are in the following figures

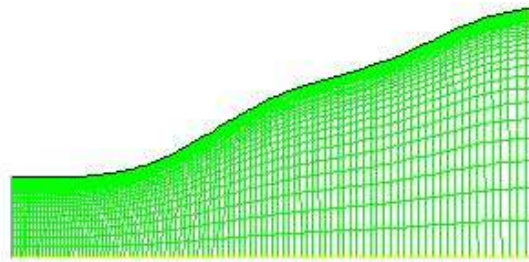


Figure 15: Mesh-5 points

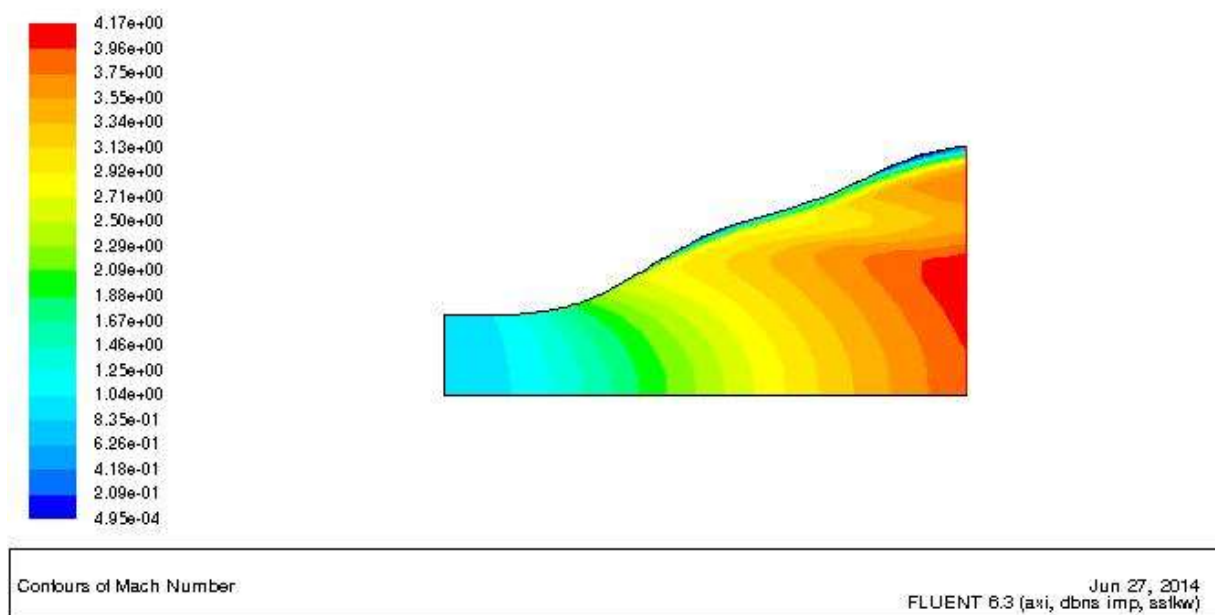


Figure 16:Mach contour- 5points

4 points with constraints on throat area

In this section results for the following condition have been described.

- 4 control point
- 1 trained network for both thrusts
- Inlet mass flow rate 20 kg/s
- With and without force on throat area

For the above condition no converged results have been obtained, however the best outputs for each cases have been shown here.

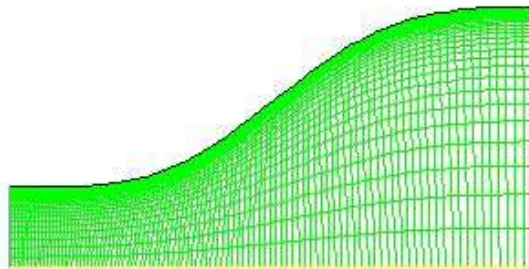


Figure 17: Grid for forced throat

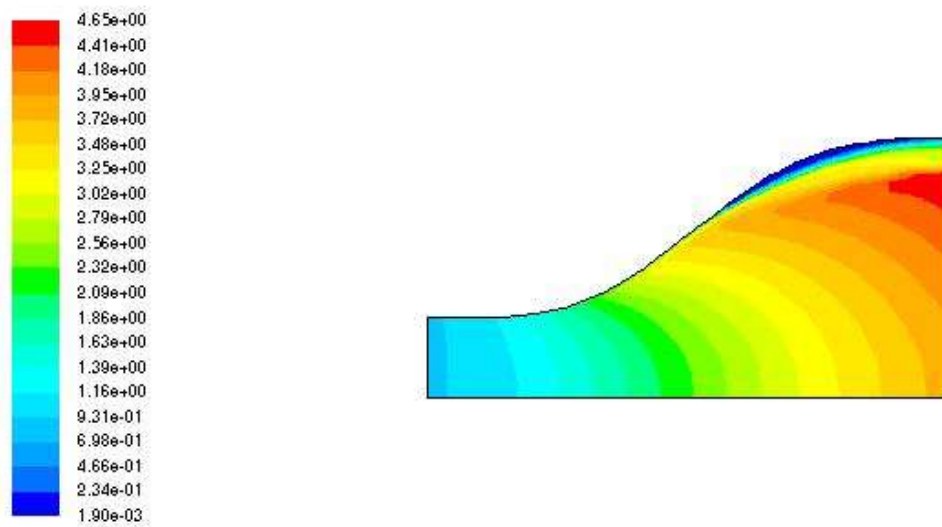


Figure 18: Mach contour for forced throat

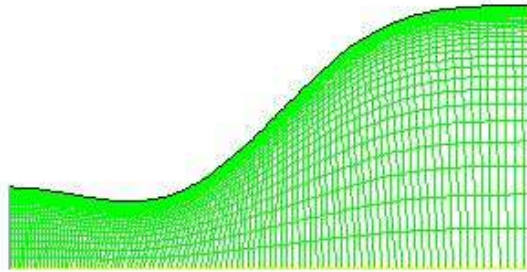


Figure 19: Grid for not forced throat

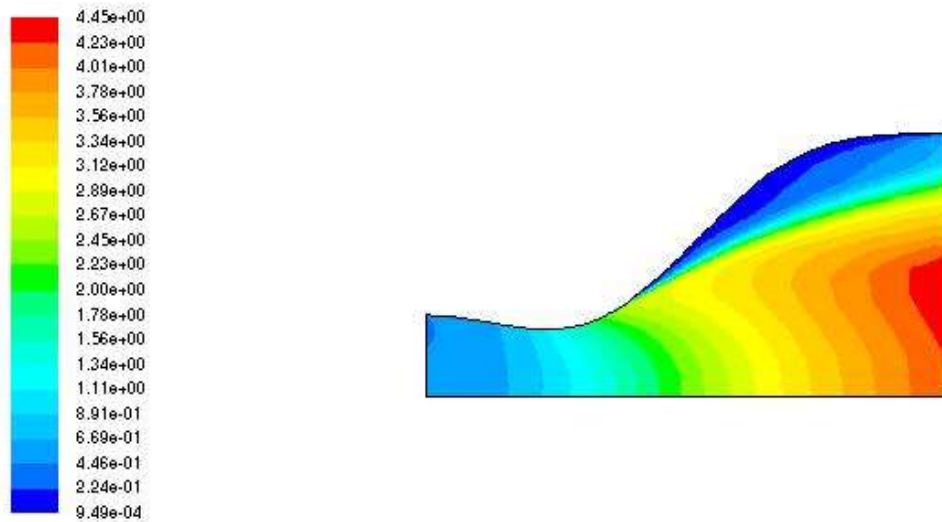


Figure 20: Mach contour for not forced throat

For the above test cases following key consideration have been achieved

- The mass flow rate for the basic points has been calculated to be 20 kg/s, however the throat condition may alter during the optimization even though the geometrical constraints have been

applied. In this way, there may be some cases that flow in the nozzle will subsonic after all. As a result, the mass flow rate has been reset to a higher value thus it is a certain that the flow will be supersonic in a major portion of nozzle.

- The network should be separated for each parts of thrust to ensure the convergence and more accurate results

Final results

For the final results following conditions are applied and the convergence achieved

- Two separated network for thrust
- 4 control points
- Constant first point
- Zero slop at the end and beginning
- The velocity thrust-pressure thrust more than 55%
- Exit Mach No. more than 2.0
- Inlet mass flow rate 30 kg/s
- No. first inputs for neural networks 15
- Final sample of neural network 110

For the above conditions the results are shown as following, the final thrust results are 37423 N consist of 13827 N velocity thrust and 23599 N pressure thrust.

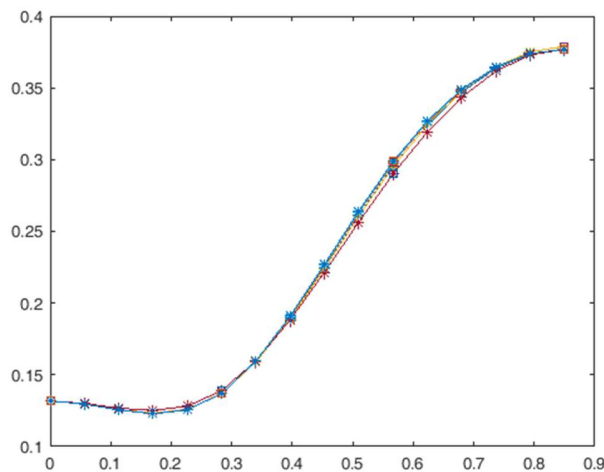
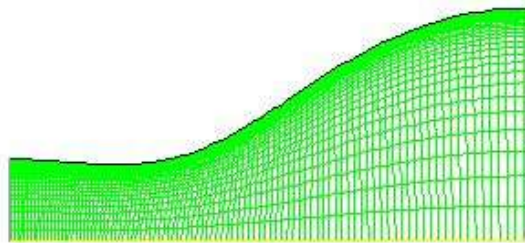


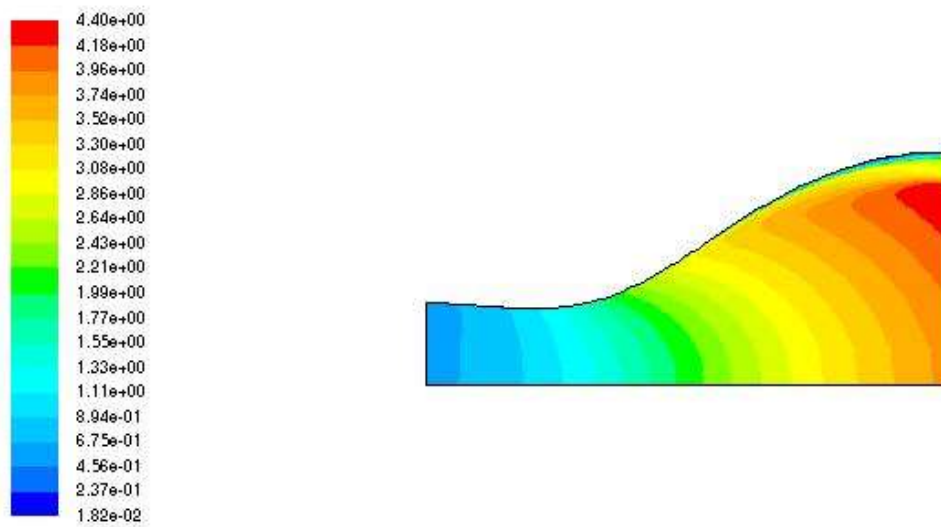
Figure 21: Final converged profiles with highest out put thrust



Grid

FLUENT 6.3 (axi, dbns imp, ss1kw) Jun 26, 2014

Figure 22:Final answer grid



Contours of Mach Number

FLUENT 6.3 (axi, dbns imp, ss1kw) Jun 26, 2014

Figure 23:Final answer Mach contour

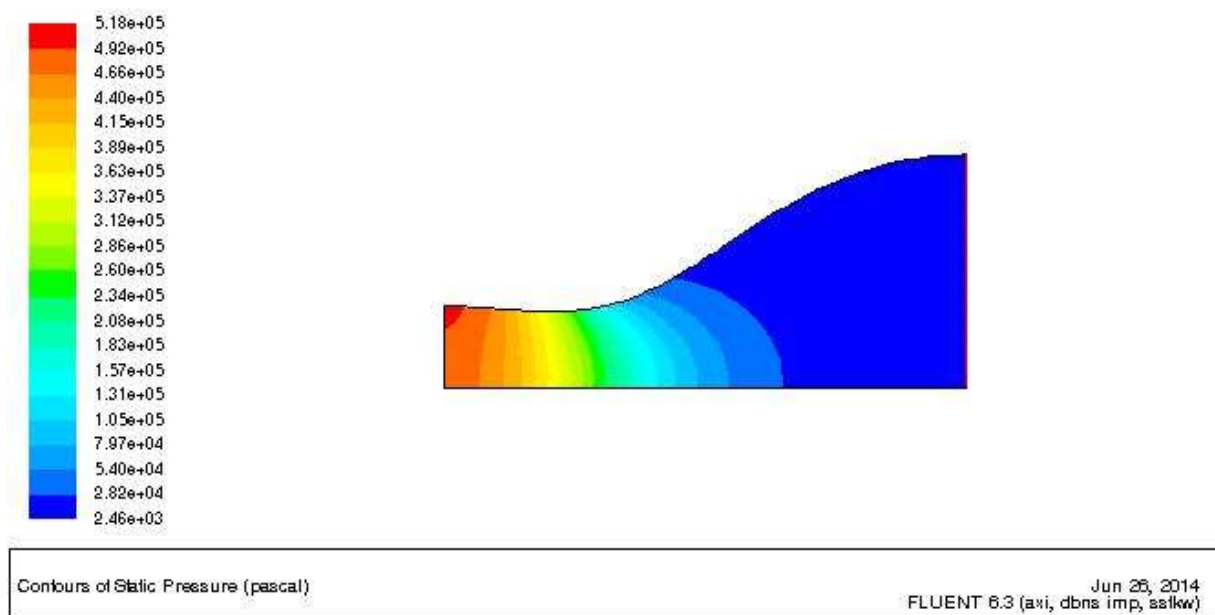


Figure 24: Final answer pressure contour

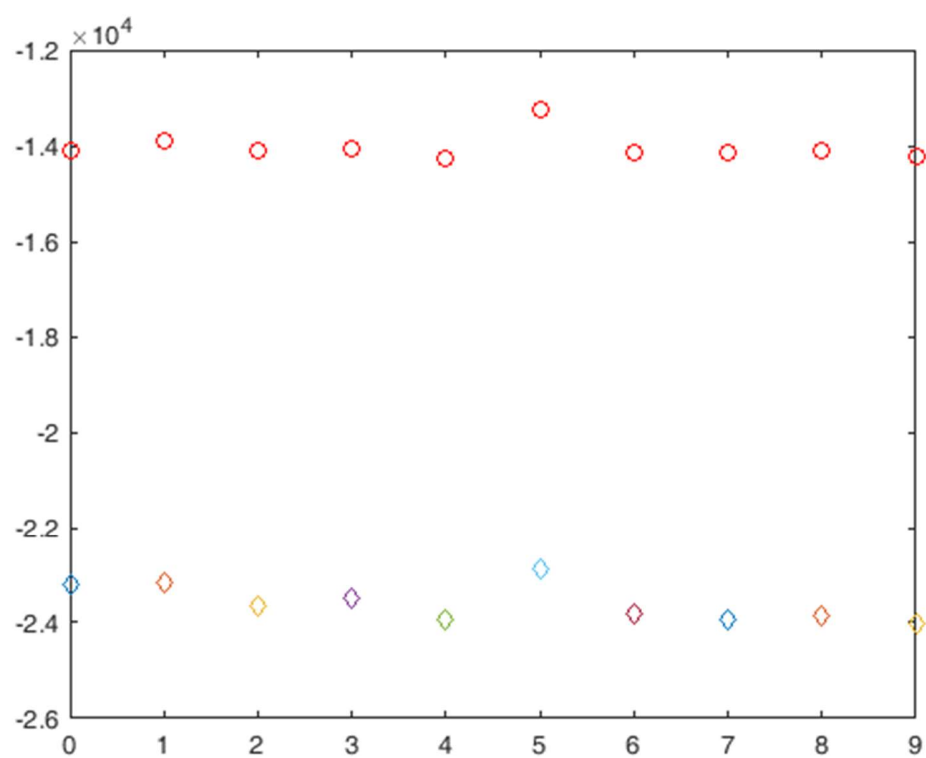


Figure 25: Final answer pressure and velocity convergence history

Conclusion

Based on comprehensive study following results

The results will be more accurate for separated network for each output variable

Chocking mass flow rate should be applied with a safety factor to ensure supersonic flow in each test cases, in other work for a fixed mass flow rate, the throat should be assumed to be smaller in order to get chocked any way

The thrust ratio constraint has been obtained based survey in output data, since the time line of the project was tight, the deviation in control points coordinate are limited and the best thrust output has been obtained with this thrust ratio. In order to obtain any arbitrary ratio, the deviation has to be assumed more widely and may the MATLAB function, *fmincon* has to be genetic algorithm

The exit Mach no. has been set the same way thrust ratio did

Using the finite difference for setting the second derivative will not give the accurate results since the distance between point are relatively high and central, forward or backward difference will result in tremendous amount of error. To avoid that the number of control point should increase so the error in difference will decreased. By the way the increase in the control point will make the optimization very difficult to converge

Since the problem include so many nonlinearities and constraints on outputs i.e. exit Mach no. and thrust ratio, the road to achieve feasible results in a timely manner was breath taking!

Appendix

Main code

```
clc
clear all
close all
[x0,y0,z0]=textread('points.dat','%s %s %s');
%converting cel format into numbers
Y0=cellfun(@str2num,y0);
X=cellfun(@str2num,x0);
Y=Y0;
%upper and lower band
% lb = [Y0(1,1)-0.1*Y0(1,1),Y0(2,1)-0.1*Y0(2,1),Y0(3,1)-0.1*Y0(3,1),Y0(4,1)-
0.1*Y0(4,1)];
% ub =
[Y0(1,1)+0.1*Y0(1,1),Y0(2,1)+0.1*Y0(2,1),Y0(3,1)+0.1*Y0(3,1),Y0(4,1)+0.1*Y0(4,1)];
T2_opt=3;
T1_opt=3;
T1_flnt=1;
T2_flnt=1;
DT1=(T1_opt-T1_flnt)/T1_flnt;
DT2=(T2_opt-T2_flnt)/T2_flnt;
N=0;
while N<10
A=[1 -1 0 0 ;0 1 -1 0 ;0 0 1 -1];

b=[0; 0; 0];
Aeq=[1 0 0 0 ];
beq=[.1318];
options = optimoptions('fmincon','MaxFunctionEvaluations',10000);
[Y] = fmincon('Thrust_opt',Y0,A,b,Aeq,beq,lb,ub)
[T1_opt,T2_opt]=Thrust_opt(Y);

S_out=point_gen_final(X,Y);
% figure(2)
% plot(X,Y)
gambit_run();
% hold on
fluent_run();
[T1_flnt,T2_flnt]=Thrust_clc(S_out);
figure(2)
plot(N,T1_flnt(end),'ro')
hold on
plot(N,T2_flnt(end),'d')
data_recorder(T1_flnt,T2_flnt,Y);
Ann_builder_tp;
Ann_builder_tv;
Thrust_ratio=T1_flnt/T2_flnt
N=N+1;
End
```

Control point and profile generation

```
function [x,y,S_out]= Cont_pt_generator()
%reading control points
[x0,y0,z0]=textread('points.dat','%s %s %s');
%converting cel format into numbers
x0=cellfun(@str2num,x0)
y0=cellfun(@str2num,y0)
z0=cellfun(@str2num,z0);
%interval for generating random number.
a=-0.999;
b=0.999;
x=x0;
%production of rand control points for each coordinate
max_dev=0.1;% max allowed deviation
y=rand(length(x0),1);
while issorted(y)==0; % ensure the ascending trend in y vector
for i=1:length(x0)
    if (i==1)
        y(i,1)=y0(i,1);
    else
        y(i,1)=y0(i,1)+0.15*((b-a).*rand(1,1)+a)*y0(i,1);
    end
end
end
%y vector has been generated
% plot(x0,y0,'s')
% hold on
% plot(x,y,'d')
% xx=0:0.01:max(x);
% yy=spline(x,y,xx);
% yy0=spline(x0,y0,xx);
% plot(xx,yy)
% hold on
% plot(xx,yy0)
% plotting x & y makes us shore that generated y is within the range
% plot(x,y,'*')
% hold on
% plot(x0,y0,'s')
S_out=3.14*y(end,1)^2;
*****
function [y,S_out]=point_gen()
if exist('xy_data.dat')==1
    delete('xy_data.dat')
end
pause(5)
[x,y,S_out]= Cont_pt_generator();
% spline_fit=fit(x,y,'poly2')
%xf=[x(1,1);linspace(x(2,1),x(6,1),14)';x(7,1)];
% yf=feval(spline_fit,xf)
zf=zeros(16,1);
%plot(x,yf,'*')
figure(1)
plot(x,y,'s')
hold on
xf=[linspace(x(1,1),x(end,1),16)];
```

```
yf=spline(x',[0 y' 0],xf);  
% %yf=interp1q(x,y,xf);  
plot(xf,yf,'-*')  
hold on  
% B=[xf;yf];  
S_out=3.14*y(end);  
fid=fopen('xy_data.dat','w');  
fprintf(fid,'%f %f %f\r\n',[xf',yf',zf]');  
fclose(fid)
```

GAMBIT AND Fluent Run

```
function [y,S_out]=gambit_run()
if exist('nozzle03.msh')==0
    delete('nozzle03.msh')
end
if exist('nozzle03.dbs')==0
    delete('nozzle03.dbs')
end
if exist('nozzle03.trn')==0
    delete('nozzle03.trn')
end
if exist('nozzle03.jou')==0
    delete('nozzle03.jou')
end
pause(5)
[y,S_out]=point_gen();

system('C:\Gambit.Inc\ntbin\ntx86\gambit.exe -r2.4.6 -inp main_gmbt_jr.jou')

*****

function fluent_run()
if exist('nozzle03.cas')==0
    delete('nozzle03.cas')
end
if exist('nozzle03.dat')==0
    delete('nozzle03.dat')
end
if exist('mdist.txt')==0
    delete('mdist.txt')
end
if exist('mach_out.txt')==0
    delete('mach_out.txt')
end
if exist('mout.txt')==0
    delete('mout.txt')
end
if exist('pout.txt')==0
    delete('pout.txt')
end
if exist('pin.txt')==0
    delete('pin.txt')
end
if exist('vout.txt')==0
    delete('vout.txt')
end
if exist('vin.txt')==0
    delete('vin.txt')
end
system('"C:\Fluent6326x64\ntbin\win64\fluent.exe" -r6.3.26 2d -t4 -wait -i
new_fluent_journal.jou')
end
```

Thrust Calculator

```
function [T1,T2]=Thrust_clc(S_out)
mdot=30;
P_amb=0;
out1 = fopen('vin.txt','r');
out2 = fopen('vout.txt','r');
out3 = fopen('pout.txt','r');
out4 = fopen('pin.txt','r');
for i=1:6
    Line1 = fgetl(out1);
    Line2 = fgetl(out2);
    Line3 = fgetl(out3);
    Line4 = fgetl(out4);
    if (i==6) %if2
        V_in = Line1(42:length(Line1));
        V_in = str2num(V_in);
        V_out = Line2(42:length(Line2));
        V_out = str2num(V_out);
        P_out = Line3(42:length(Line3));
        P_out = str2num(P_out);
        P_in = Line4(42:length(Line4));
        P_in = str2num(P_in);
    end %if2
end
fclose('all')
T1=-mdot *( V_out- V_in)
T2=- (P_out-P_amb)*S_out
```

data base for neural net works

```
function [T1,T2,y]=output_loop(N)
for i=1:N
    [yt(:,1,i),S_out]=gambit_run();
    fluent_run();
    [T1(i),T2(i)]=Thrust_clc(S_out);
    pause(5)

    % xlswrite('Thrust_database.xlsx',y,'sheet2',)
end
for i=1:N
    for j=1:4
        y(j,i)=yt(j,1,i);
    end
end
for i=1:N
    T1(1,i)=T1(i);
    T2(1,i)=T2(i);
end
xlswrite('xy_data.xlsx',y,'sheet1')
xlswrite('Thrust_database.xlsx',T1,'sheet1')
xlswrite('Thrust_database.xlsx',T2,'sheet2')
end
```

pressure and velocity thrust networks

```
Inputs=xlsread('xy_data.xlsx','sheet1');
Tv=xlsread('Thrust_database.xlsx','sheet1');
Target=[Tv];
NNeron = [5 5 5];
mynetv = newcf(Inputs,Target,NNeron);
mynetv.trainParam.epochs = 500;
mynetv.trainParam.goal = 1e-16;
```

```
%% Training The mynetv
mynetv= train(mynetv,Inputs,Target);
Outputs = mynetv(Inputs);
perf = perform(mynetv,Outputs,Target);
save('ANN_tv','mynetv')
```

```
*****
```

```
Inputs=xlsread('xy_data.xlsx','sheet1');
Tp=xlsread('Thrust_database.xlsx','sheet2');
Target=[Tp];
NNeron = [5 5 5];
mynetp = newcf(Inputs,Target,NNeron);
mynetp.trainParam.epochs = 500;
mynetp.trainParam.goal = 1e-16;
```

```
%% Training The mynetp
mynetp= train(mynetp,Inputs,Target);
Outputs = mynetp(Inputs);
perf = perform(mynetp,Outputs,Target);
save('ANN_Tp','mynetp')
```

Pressure and velocity thrust calculator

```
function [T_p]=Thrust_press(Y)
% ann_net=load('mynet.mat')
load('ANN_tp','mynetp');
T_p=sim(mynetp,Y);
% size(TT)
```

```
function [T_v]=Thrust_velocity(Y)
% ann_net=load('mynet.mat')
load('ANN_tv','mynetv');
T_v=sim(mynetv,Y);
% size(TT)
```


Objective thrust function

```
function [T1,T2]=Thrust_opt(Y)
% ann_net=load('mynet.mat')
load('ANN_tv','mynetv');
load('ANN_tp','mynetp');
T_p=sim(mynetp,Y);
T_v=sim(mynetv,Y);
% size(TT)
T1=T_v;
T2=T_p;
```

Updating network database

```
function data_recorder(T1_flnt,T2_flnt,Y)
Inputs=xlsread('xy_data.xlsx','sheet1');
T1=xlsread('Thrust_database.xlsx','sheet1');
T2=xlsread('Thrust_database.xlsx','sheet2');
[m,n]=size(T1);
TT1=zeros(1,n+1);
for i=1:n
    TT1(1,i)=T1(1,i);
end
TT1(1,n+1)=T1_flnt;
xlswrite('Thrust_database.xlsx',TT1,'sheet1');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,n]=size(T2);
TT2=zeros(1,n+1);
for i=1:n
    TT2(1,i)=T2(1,i);
end
TT2(1,n+1)=T2_flnt;
xlswrite('Thrust_database.xlsx',TT2,'sheet2');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m,n]=size(Inputs)
Inp=zeros(m,n+1);
for i=1:m
    for j=1:n
        Inp(i,j)=Inputs(i,j);
    end
end
Inp(:,n+1)=Y(:,1);
xlswrite('xy_data.xlsx',Inp,'sheet1');
```

Thrust ratio Constraints

```
function [c,ceq] = constraints(Y)
c1=Thrust_press(Y);
c2=Thrust_velocity(Y);
g1=c1/c2-0.60;
c =[g1];
ceq=[];
```

Sample Project For CeADAR