AE 5335 is taught by Dr. Riggins

Final Project

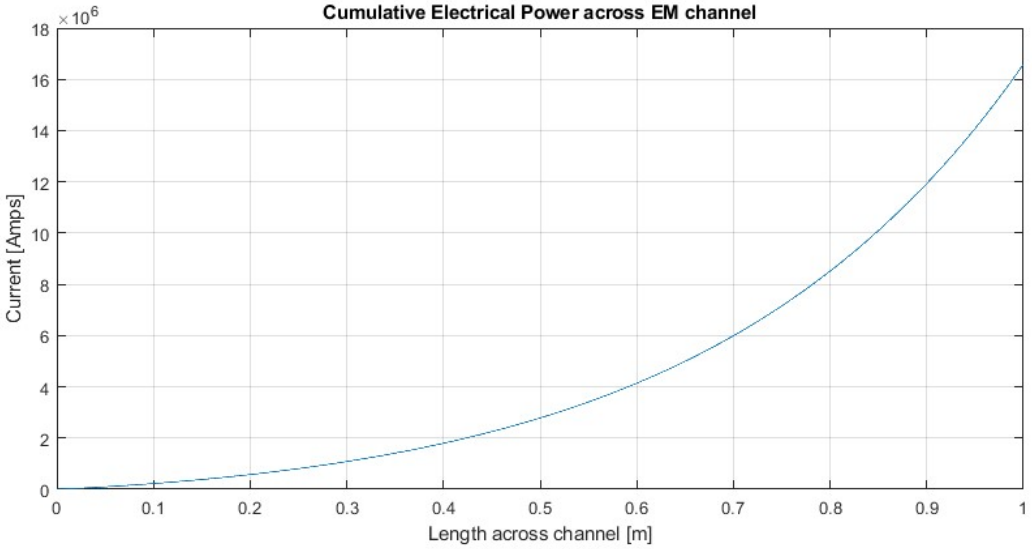
Propulsion 2

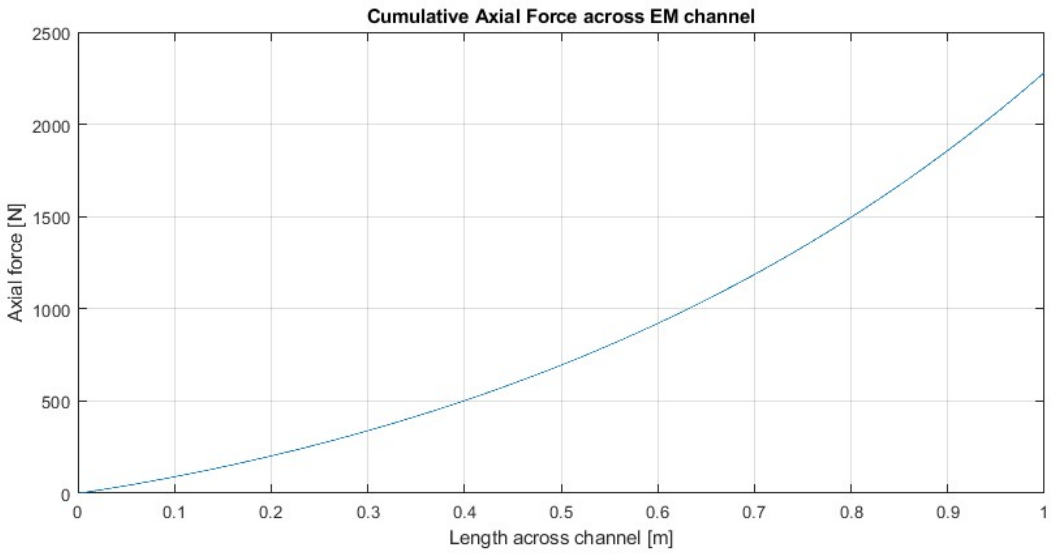
Matt Pahayo

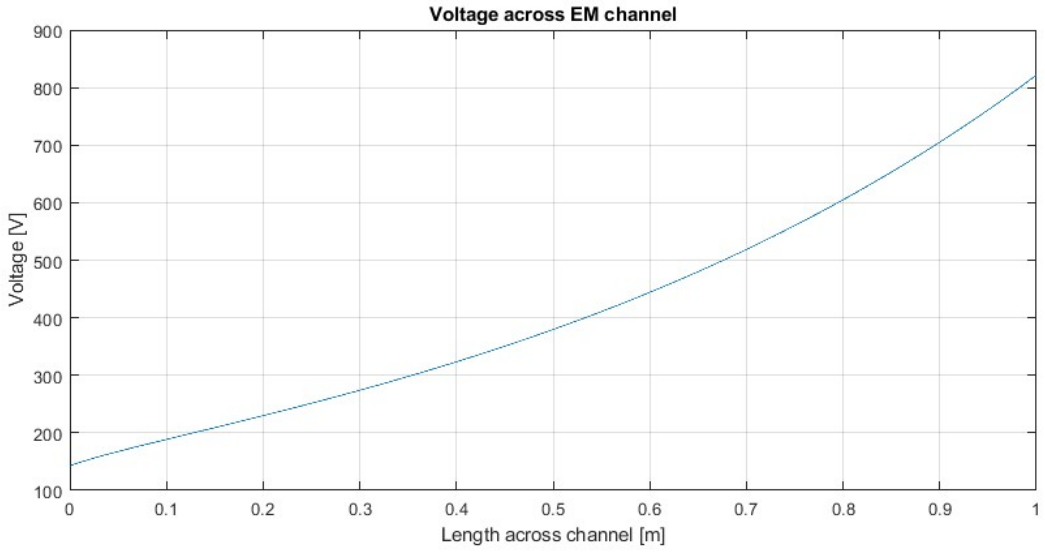
1.0 Results

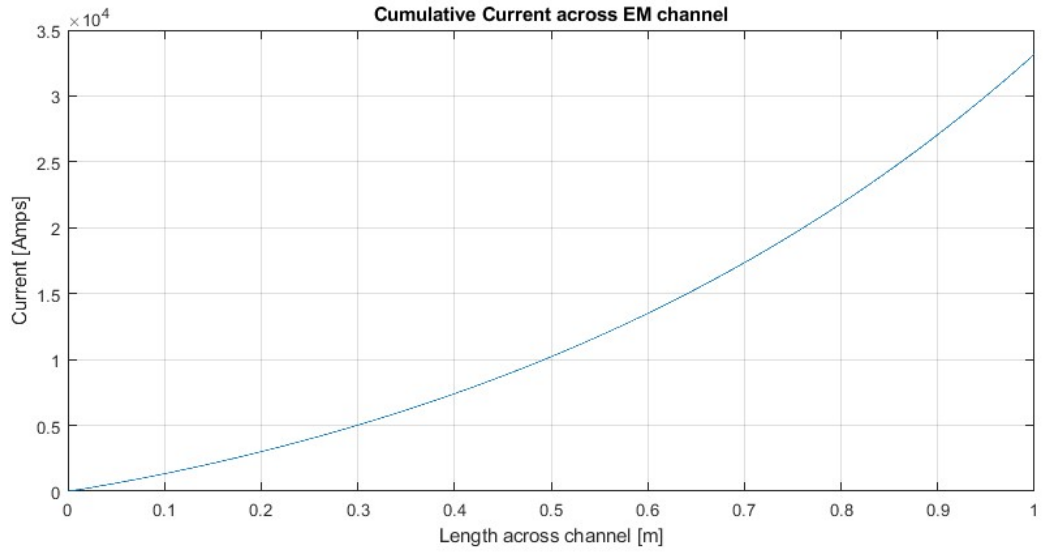
Fluid Dynamics			
x	Inlet of channel	exit of channel	exit of nozzle
u [m/s]	1631.293965	9388.876206	10596.17541
Mach	1.2	3.938143876	11.43758172
T [K]	320	984.2197599	148.6202101
T_t [K]	412.16	4037.06812	4037.06812
P [Pa]	100000	53447.75804	71.5121369
Pt [Pa]	242496	7470307.593	7470307.593

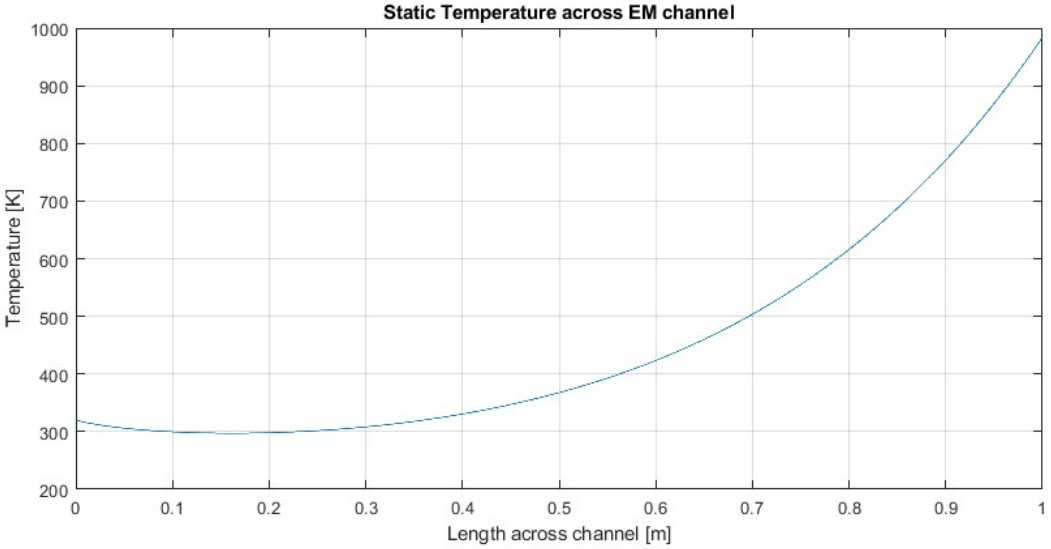
Requirements			
B [T]	1.4	Thrust [N]	3275.002102
σ_0 [mho]	400	Isp [sec]	1080.140205
η	0.8	mdot [kg/s]	0.308957
payload mass [kg]	500	electrical power rate [kW]	16593.87356
engine mass [kg]	4976.64	heat rate [kW]	-424.752
propellant mass [kg]	8595.944039		
initial mass [kg]	14072.58404		

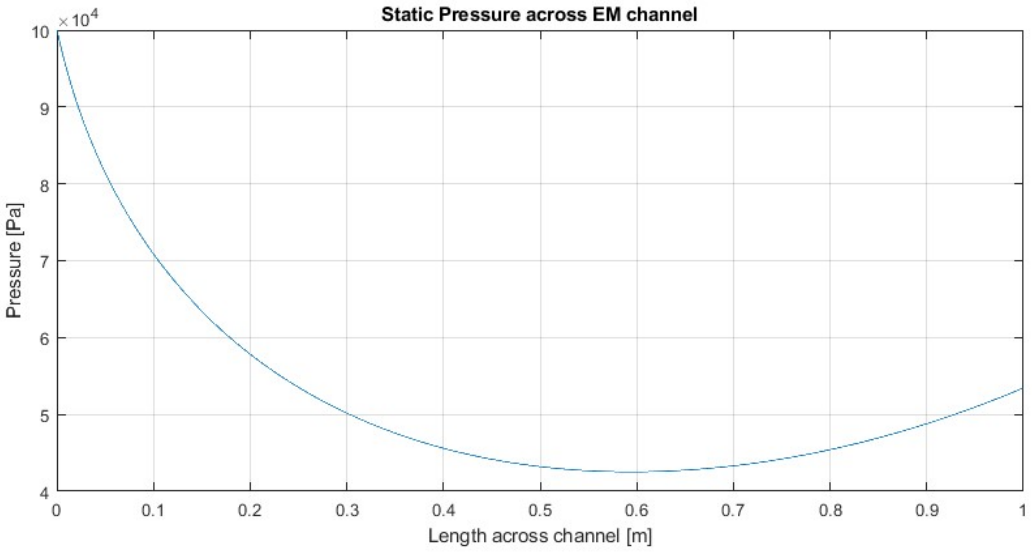


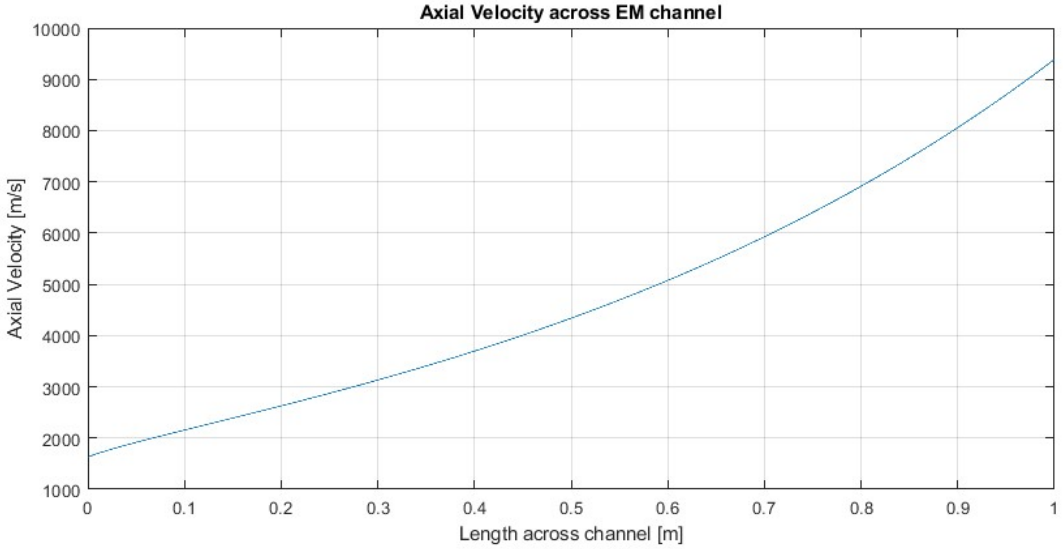

Cumulative Axial Force across EM channel
Axial force [N]
Length across channel [m]
2500
2000
1500
1000
500
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

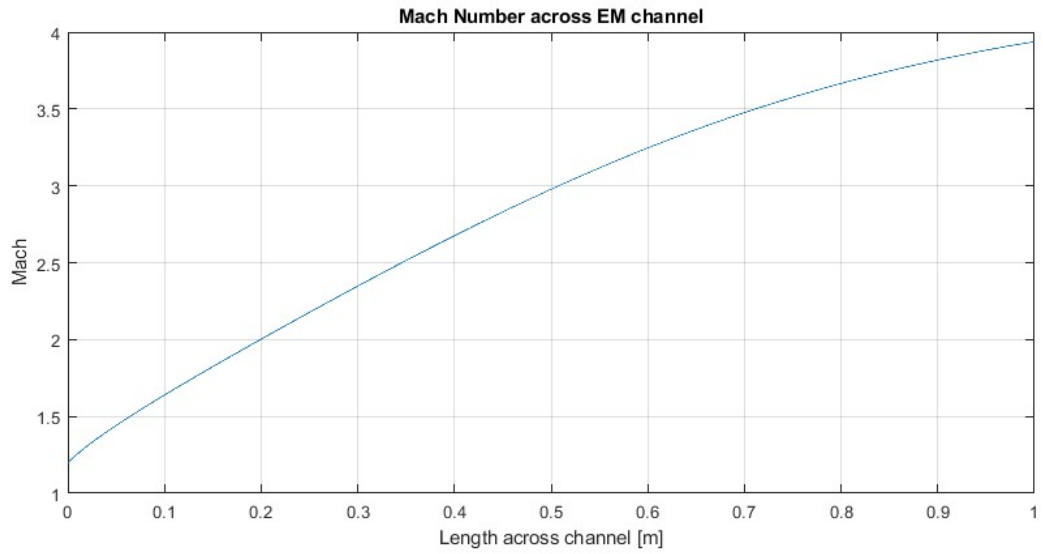

Voltage across EM channel
Voltage [V]
Length across channel [m]
900
800
700
600
500
400
300
200
100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

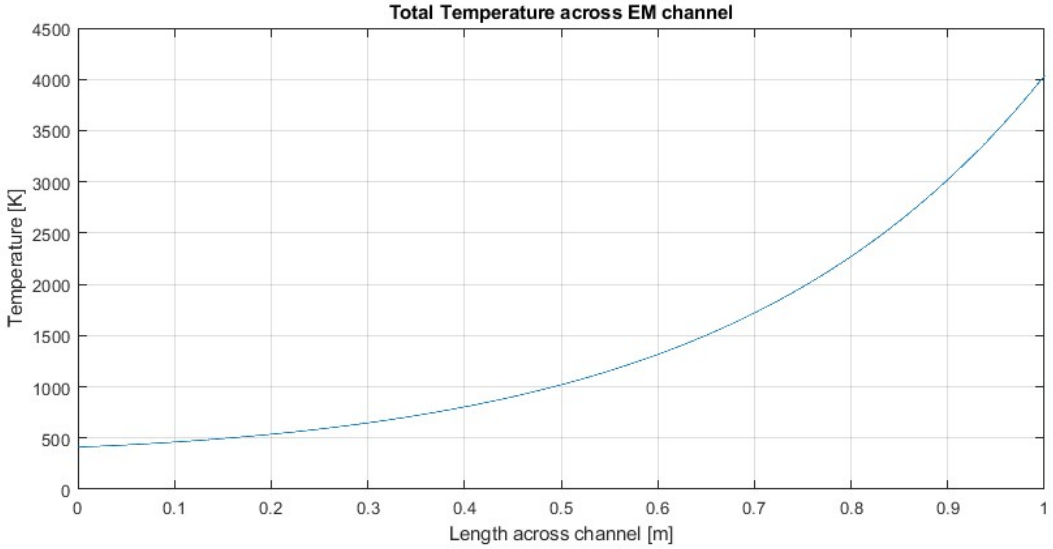

Cumulative Current across EM channel
×10^4
3.5
3
2.5
2
1.5
1
0.5
0
Current [Amps]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Length across channel [m]

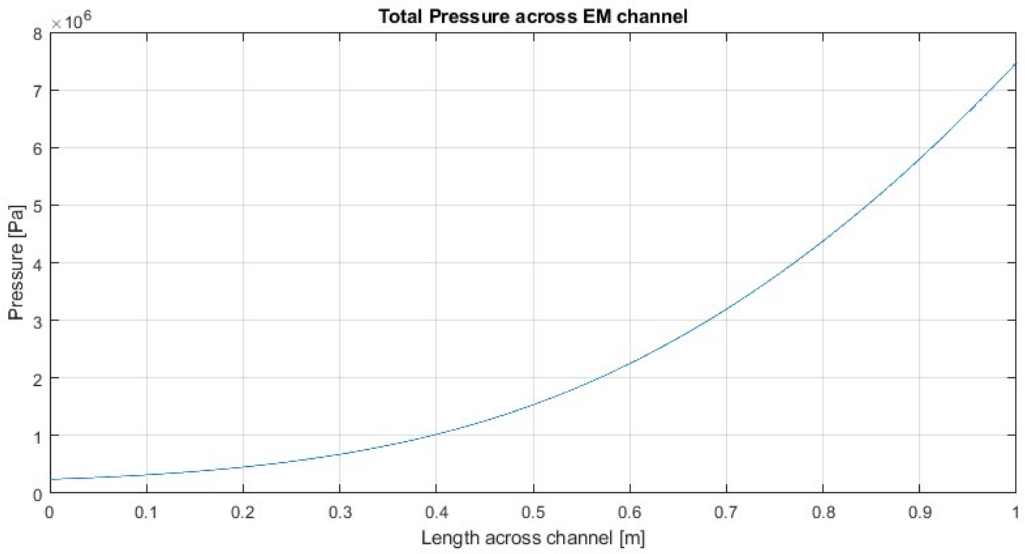

Static Temperature across EM channel
1000
900
800
700
600
500
400
300
200
Temperature [K]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Length across channel [m]


Static Pressure across EM channel
×10^4
10
9
8
7
6
5
4
Pressure [Pa]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Length across channel [m]


Axial Velocity across EM channel
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
Axial Velocity [m/s]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Length across channel [m]


Mach Number across EM channel
Mach
Length across channel [m]
4
3.5
3
2.5
2
1.5
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Total Temperature across EM channel
Temperature [K]
Length across channel [m]
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



2.0 Methodology

First off, the problem is to solve a non-linear set of differential equations. The method that will be used to solve the non-linear set is a Newton-Raphson method for multivariable systems.

$$F\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}\right) = \{f\} = \begin{bmatrix} f_1(x_1, x_2, x_3, x_4) = 0 \\ f_2(x_1, x_2, x_3, x_4) = 0 \\ f_3(x_1, x_2, x_3, x_4) = 0 \\ f_4(x_1, x_2, x_3, x_4) = 0 \end{bmatrix}$$

In this case or functions are the differential equations for continuity, momentum, energy, and the equation of state.

$$f_1 = \frac{d\rho}{\rho} + \frac{du}{u} + \frac{dA}{A} = 0$$

$$f_2 = \frac{dP}{\rho} + udu + \frac{\tau_w c dx}{\rho A} - \eta\delta w = 0$$

$$f_3 = C_p dT + udu - \delta q - \delta w = 0$$

$$f_4 = \frac{dP}{P} - \frac{d\rho}{\rho} + + \frac{dA}{A} = 0$$

The Jacobian is needed for further calculation and is denoted [J].

$$[J] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix}$$

$$[J]^k\{\Delta x\}^k = -\{f\}^k \quad (1)$$

At the first iteration {x} is the values at the inlet. To get the solution vector for equation 1, use an algorithm for solving linear equations. It was chosen that a Gauss elimination algorithm ought to be used. Alternatively, the built-in function in MATLAB may be used instead (linsolve uses LU factorization algorithm).

After solving equation 1, we can get the next value of {x}.

$$\{\Delta x\}^{k+1} = \{\Delta x\}^k + \{\Delta x\}^k \quad (2)$$

The exit criteria are the L2 norm of the current iteration of {f} scaled with the L2 norm of {f} at the original guess/inlet conditions. To exit this must be less than a tolerance value (ε). Epsilon was chosen as 10^-5. By decreasing epsilon more accurate results are found at the expense of computation time.

$$\frac{\|\{f\}^k\|}{\|\{f\}^0\|} < \epsilon$$

The amount that the steps are preformed are dependent on the step size used. Since a step size of 10000 was used, the number of times the Newton-Raphson algorithm is 10000. For each step, the {x} vector found at the end of the previous step is the initial guess of the next step.

To find the optimum values of B, sigma0, and eta to minimize the initial mass. A brute force method was used. There are 75 different combinations; at each combination of B, sigma0, and eta was put in to the MHD solver with 100 axial steps. For each combination initial mass was tabulated also total temperature was tabulated. The tabulated results will be given in the appendix. A minimum value of the initial mass was chosen with the constraint that the total temperature must not exceed 6000 K.

$$\frac{d\rho}{\rho} + \frac{du}{u} + \frac{dA}{A} = 0$$

$$\frac{dp}{\rho} + u\,du = -\frac{\tau_w\, e\, dx}{\rho A} \cdot G\left(\frac{1}{\eta} - 1\right)$$

$$c_p\,dT + u\,du = \delta q_{conv} + G(1/\eta)\left(\frac{1}{\eta} - 1\right)$$

$$\frac{dp}{p} = \frac{d\rho}{\rho} + \frac{dT}{T}$$

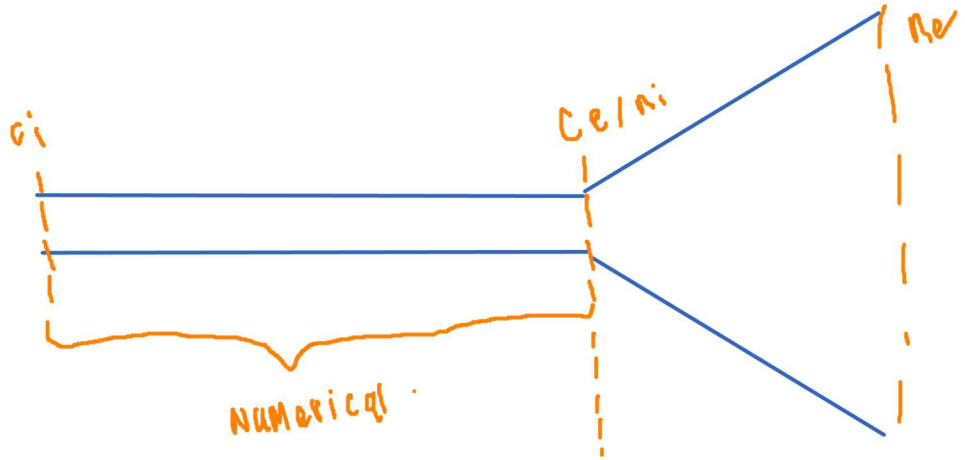
$$\dot{m}\left[G(1/\eta)\left(\frac{1}{\eta} - 1\right)\right] = \frac{Bu}{\eta} h \times j w dx$$

$$\frac{\delta q_{conv}}{c_p T_t} = \frac{N_{St}\,(T_w - T_t)}{T_t}\,\frac{C}{A}\,dx$$

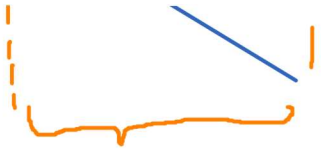
$$\delta q_{conv} = \frac{c_p c_f (T_w - T_t)}{2}\,\frac{C}{A}\,dx \qquad \frac{2 c_p C_f (T_w - T_t)\,\frac{\pi D}{\left(\frac{\pi D^2}{4}\right)}}{}$$

$$\frac{A_2}{A_1}\,G(M_1^2)$$

$$G(M_1^2) = \frac{1}{M_1}\left[\frac{2}{\gamma+1}\left(1 + \frac{\gamma-1}{2} M_1^2\right)\right]^{\frac{\gamma+1}{2(\gamma-1)}} = 1.16$$



Numerical



Analytical

$M_{ci} = 1.2$

$P_{ci} = 100\ kpa$

$T_{ci} = 320\ K$

$T_w = 220\ k$

$C_f = .002$

$A = .0625\ m^2$

$l = 1\ m$

$\frac{A_{ne}}{A_{ni}} = 100$

$M_L = 500\ kg$

$$T_{+ci} = T_{ci}\left[1 + \frac{\gamma - 1}{2} M_{ci}^2\right] = 412.16\ K$$

$$P_{+ci} = 242496\ N/m^2$$

$$\dot{m} = \frac{P}{RT} M \sqrt{\gamma R T}\, A = 0.308957\ kg/s$$

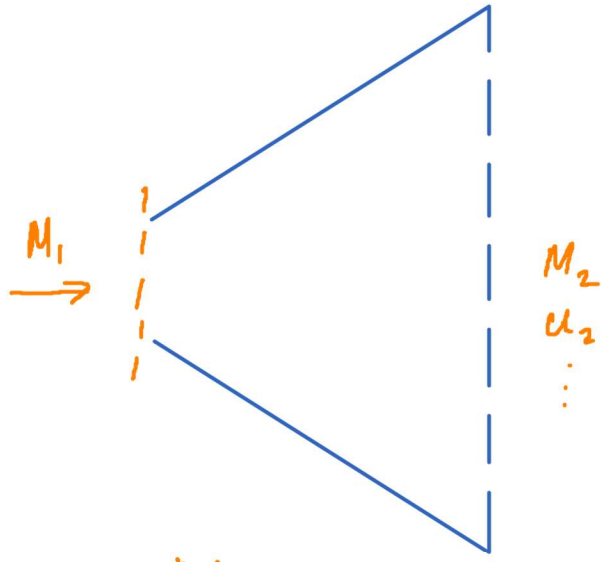
code

- M_{ce} = 9388.87620582454
- M_{ce} = 3.93814387583594
- T_{ce} = 984.219759850445
- T_+ = 4037.06812031099
- P = 53447.7580351589
- P_+ = 7470307.59327561

B = 1.4 T
Sig0 = 400 mho
Eta = 0.8

code

$$G(M_1)^2 \frac{A_{Nt}}{A_{cr}} = G(M_2^2)$$



$$G(M_1^2) = M_1^{-1}\left[\frac{2}{\gamma+1}\left(1+\frac{\gamma-1}{2}M_1^2\right)\right]^{\frac{\gamma+1}{2(\gamma-1)}} = 10.1411376183559$$

$$\Rightarrow G(M_2^2) = 1014.11376183559 \Rightarrow M_2 = 11.4375817247634$$

$$T_2 = \frac{T_t}{\left(1+\frac{\gamma-1}{2}M_2^2\right)} = \frac{4037.06812031099}{1+.2\,M_2^2} = 148.620210102\ K$$

$$P_2 = \frac{P_t}{\left(1+\frac{\gamma-1}{2}M_2^2\right)^{\gamma/\gamma-1}} = 71.5121369045636\ pa$$

$$u_2 = M\sqrt{\gamma R T_2} = 10596.1754148476$$

$$Thrust = \dot{m}u_e + P_e A_e = 0.309057107275532*10596.1754148476+71.5121369045636*.05*.05 = 3275.002102239171\ N$$

$$I_{sp} = \frac{c}{g_0}\ ;\ c \approx u_e \Rightarrow I_{sp} = 10596.1754148476/9.81 = 1080.1402053871\ sec$$

$$m_F = m_e + m_L$$

$$m_e = 4976.64\ kg$$

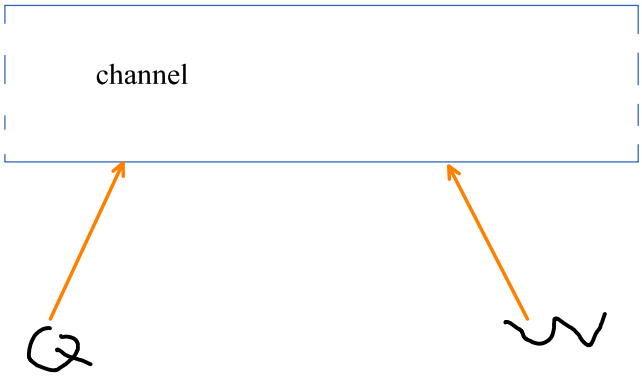
$$m_p \Rightarrow \Delta V = V_e \ln\frac{m_F + m_p}{m_F}$$

$$m_0 = m_p + m_p$$

$$\left(e^{\frac{\Delta V}{V_e}}\right) * m_F - m_F = m_p = 8595.94403928879\ kg$$

$$\dot{m}(h_2 - h_1) = W_{elec} + Q$$

$\dot{m}(h_2 - h_1) = W_{elec} + Q$



$\delta W = VI$

$W \int_0^t \delta W \Rightarrow$

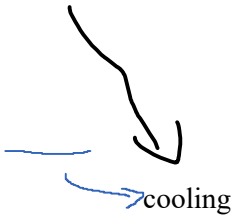
$\dot{m} C_p (T_{+2} - T_{+1}) =$

$-W - \dot{m}(h_{+2} - h_{+1}) = Q$ =16169120.6650134 -16593873.561485 kW = --424752.896471599 W = -424.752 kW

16593873.561485 W = 16593.873561485 kW

0.309057107275532*14437.5(4037.06812031099-412.16)=

16169120.6650134 W =16169.1206650134 kW



3.0 Appendix

3.1 IM.m

```
% Iterative Methods class
% used to solve linear and non-linear systems iteratively
classdef IM
    methods (Static)
%=========================================================================%
                      % Gauss-Seidel Method
%=========================================================================%
function [x,w] = gauSei(A,b,n,x,imax,es,lambda)
    for i = 1:n
        dum = A(i,i);
        for j = 1:n
            A(i,j) = A(i,j)/dum;
        end
        b(i) = b(i)/dum;
    end
    for i = 1:n
        sum = b(i);
        for j = 1:n
            if i~= j
                sum = sum - A(i,j)*x(j);
            end
            x(i) = sum;
        end
    end
    iter = 1;
    sen = 0;
    L2norm_0 = norm(b-A*x);
    while sen == 0
        sen = 1;
        for i = 1:n
            old = x(i);
            sum = b(i);
            for j = 1:n
                if i~= j
                    sum = sum - A(i,j)*x(j);
                end
            end
            x(i) = lambda*sum + (1-lambda)*old;
            L2norm = norm(b-A*x);
            if sen == 1 && x(i) ~= 0
                ea = abs(L2norm/L2norm_0)/1;
                if ea > es
                    sen = 0;
                end
            end

        end
        iter = iter + 1;
        if iter >= imax
```

```
            break
        end
    end
    w = [lambda iter];
end
%=========================================================================%
                      % Newton-Raphson Method
%=========================================================================%
function [q] = newRap(f,q,p,kmax)
    % f is the 'A' matrix
    % q is the 'b' vector
    % p is the precision goal
    % kmax is the maximum allowable iterations
    syms x1 x2 x3 x4
    fp(x1,x2,x3,x4) = jacobian(f,[x1 x2 x3 x4]);
    b = transpose(double(f(q(1),q(2),q(3),q(4))));
    b_0 = b;
    k = 0;
     while (norm(b)/norm(b_0)) > 10^p && k<kmax
        A = double(fp(q(1),q(2),q(3),q(4)));
        b = transpose(double(f(q(1),q(2),q(3),q(4))));
        del = gauss(A,-b); % gauss elimination algorithm
        q = q+del;
        k = k + 1;
     end
end

    end
end
```

3.1 Gauss elimination algorithm

```
function [x] = gauss(a,b)
% gauss elimination

n = length(a);

k = 1 ;
p = k ;
big = abs(a(k,k));

%**************************************************
% pivoting portion
%**************************************************
for ii=k+1:n
    dummy = abs(a(ii,k));
    if dummy > big
       big = dummy;
       p = ii ;
    end
end
if p ~= k
    for jj = k:n
        dummy = a(p,jj);
        a(p,jj) = a(k,jj);
```

```
        a(k,jj) = dummy;
    end
    dummy = b(p);
    b(p)=b(k);
    b(k) = dummy;
end

%*************************************************
% elimination step
%*************************************************
for k=1:(n-1)
    for i=k+1:n
        factor = a(i,k)/a(k,k);
       for j=k+1:n
           a(i,j) = a(i,j) - factor*a(k,j);
       end
       b(i) = b(i) - factor*b(k);
    end
end

%*************************************************
% back substitution
%*************************************************
x(n,1) = b(n)/a(n,n);
for i = n-1:-1:1
    sum = b(i);
   for j = i + 1:n
       sum = sum - a(i,j)*x(j,1);
   end
   x(i,1) = sum/a(i,i);
end
end
```

3.2 main.m

```
clc
clear all
close all

height = .05;
width = .05 ;
steps = 100;
cf = 0.002;
eta = .8;
Tw = 220;
w = 0;
h = 0;
A = ones(1,steps)*height*width;
% for convective heat tranmdot0sfer set ht to 1 else set it to 0
ht = 1;
l = 1;
sig0 = 400;
B = 1.4;
R = 4125;
```

```
gam =1.4;


M = 1.2
Pi = 100000
Ti = 320

[P,T,Pt2,Tt2,rho,u,M,thrust,F,mdot0,mdot,V,Ic,Powerc,forcec] =
MHD(Pi,Ti,M,cf,Tw,eta,w,h,ht,A,l,sig0,B,steps,R,gam,height,width);


Gm1 = 1/M(end)*(2/(gam+1)*(1+(gam-1)/2*M(end)^2))^((gam+1)/(2*(gam-1)))
Gm2 = Gm1 * 100

syms X
func = symfun(1/X*(2/(gam+1)*(1+(gam-1)/2*X^2))^((gam+1)/(2*(gam-1)))-Gm2,X)
Me = rootFind.newRap(func,7)
Te = Tt2(end)/(1+(gam-1)/2*Me^2)
ue = Me*sqrt(gam*R*Te)


ml = me(B,sig0,eta)
mp = exp(10000/ue)*(ml+500)-(ml+500)
m0 = ml+500+ mp
Tt2(end)

function m_e = me(B,sig0,eta)
    N_B = (B-.6)/.2;
    N_sig0 = (sig0-100)/100;
    N_eta = (1-eta)/.1;
    m_e = 1000*1.2^(N_B-1)*1.2^(N_sig0-1)*2^(N_eta-1);
end
```

3.4

```
% initialMass.m
clc
clear all
close all
y = 1
for i = 1:5
    sig0 = i*100+100;
    for j = 1:5
        B = j*.2+.6;
        for k = 1:3
            eta = k*.1+.6;
            height = .05;
            width = .05 ;
            steps = 100;
            cf = 0.002;
%             eta = .9;
            Tw = 220
            w = 0;
```

```
            h = 0;
            A = ones(1,steps)*height*width;
            % for convective heat transfer set ht to 1 else set it to 0
            ht = 1;
            l = 1;
%             sig0 = 600;
%             B = .8;
            R = 4125;
            gam =1.4;


            M = 1.2;
            Pi = 100000;
            Ti = 320;

            [P,T,Pt2,Tt2,rho,u,M,thrust,F,mdot0,mdot,V,Ic,Powerc,forcec] =
MHD(Pi,Ti,M,cf,Tw,eta,w,h,ht,A,l,sig0,B,steps,R,gam,height,width);


            Gm1 = 1/M(end)*(2/(gam+1)*(1+(gam-
1)/2*M(end)^2))^((gam+1)/(2*(gam-1)))
            Gm2 = Gm1 * 100

            syms X
            func = symfun(1/X*(2/(gam+1)*(1+(gam-1)/2*X^2))^((gam+1)/(2*(gam-
1)))-Gm2,X)
            Me = rootFind.newRap(func,7)
            Te = Tt2(end)/(1+(gam-1)/2*Me^2)
            ue = Me*sqrt(gam*R*Te)


            ml = me(B,sig0,eta)
            mp = exp(10000/ue)*(ml+500)-(ml+500)
            m0(y,1) = ml+500+ mp
            maxTt(y,1)=Tt2(end)
            s(y,1)=sig0
            b(y,1)=B
           e(y,1)= eta
            y = y+1
        end
    end
end

function m_e = me(B,sig0,eta)
    N_B = (B-.6)/.2;
    N_sig0 = (sig0-100)/100;
    N_eta = (1-eta)/.1;
    m_e = 1000*1.2^(N_B-1)*1.2^(N_sig0-1)*2^(N_eta-1);
end
```

3.5 MHD.m

```
function [Pv,Tv,Pt2,Tt2,rho,uv,Mv,thrust,F,mdot0,mdot,V,Ic,Powerc,forcec] =
MHD(P0,T0,M,cf,Tw,eta,w,h,ht,A,l,sig0,B,steps,R,gam,height,width)
% Author: Matt P
%{
```

```
MHD.m must include IM.m and gauss.m

P0 is the static pressure at the inlet
T0 is the static temperature at the inlet
M is the Mach # at the inlet
cf is the coefficient of skin friction
Tw is the wall temperature
eta is the thermodynamic efficiency
w is the work interaction per mass
h is the heat interaction per mass
ht is for convective heat transfer - 1 if present, 0 if not
A is the area along the length of the channel/nozzle and is a ROW VECTOR
l is the length of the device
%}

format longg

syms x1 x2 x3 x4
% x1 is P; x2 is rho; x3 is T; x4 is u
p = -7.0;
kmax = 1000;
T = T0;
P = P0;
cp = R*gam/(gam-1);
rho = P/R/T;
rho0 = rho;
M0 = M;
u = M*sqrt(gam*R*T);
u0 = u;
mdot = rho*u*A(1);
totalPower = 0;
Pv = zeros(steps-1,1);
rhov = zeros(steps-1,1);
Tv = zeros(steps-1,1);
uv = zeros(steps-1,1);
Mv = zeros(steps-1,1);
V = zeros(steps-1,1);
I = zeros(steps-1,1);
Power = zeros(steps-1,1);
F = zeros(steps-1,1);
Tt2 = zeros(steps-1,1);
Pt2 = zeros(steps-1,1);
Ic = zeros(steps-1,1);
Powerc = Ic;
forcec = Ic;
Pv(1,1)=P;
rhov(1,1)=rho;
Tv(1,1)=T;
uv(1,1)=u;
Mv(1,1)=M;
Pow = 0;
F = 0;
Icumulative = 0;
for i = 2:length(A)-1
    G = A(i)*sig0*B^2*u^2*l/length(A)/mdot;
    c = 2*width+2*height;
```

```
    f = @(x1,x2,x3,x4) ([(x2-rho)/x2+(x4-u)/x4+(A(i+1)-A(i))/A(i) (x1-
P)/x2+x4*(x4-u)+1/2*cf*rho0*u0^2*c*(l/length(A))/rho/A(i)-G*(1/eta-1)-
eta*w/length(A) ...
        cp*(x3-T)+x4*(x4-u)-ht*1/2*cp*cf*(Tw-T*(1+(gam-
1)/2*M^2))*c/A(i)*(l/length(A))-G*(1/eta-1)*1/eta-(h/length(A))-(w/length(A))
(x3-T)/x3+(x2-rho)/x2-(x1-P)/x1]);
    q = transpose([P,rho,T,u]);
    [q] = IM.newRap(f,q,p,kmax);
    P = q(1);
    rho = q(2);
    T = q(3);
    u = q(4);
    M = u/sqrt(gam*R*T);
    Pv(i,1)=P;
    rhov(i,1)=rho;
    Tv(i,1)=T;
    uv(i,1)=u;
    Mv(i,1)=M;
    F = Pv(i)/R/T*uv(i,1)*A(i)*(uv(i,1)-uv(i-1,1))+Pv(i)*A(i)-Pv(i-1)*A(i-1)
+ F;
    forcec(i-1,1) = F;
    V(i-1,1) = B*u/eta*height;
    I(i-1,1) = sig0*(B*u/eta-B*u)*width*(l/length(A));
    Icumulative = Icumulative + I(i-1,1);
    Ic(i-1,1) = Icumulative;
    Power(i,1) = V(i-1,1)*I(i-1,1);
    totalPower = totalPower + Power(i,1)
    Powerc(i-1,1) = totalPower;
    Pow = (G*(1/eta-1)*1/eta*mdot + Pow)
    Tt2(i,1) = T*(1+(gam-1)/2*M^2);
    Pt2(i,1) = P*(1+(gam-1)/2*M^2)^(gam/(gam-1));
    md = rho*u*A(i)

end

thrust = P/R/T*u*A(end)*(u-u0)+P*A(end)-P0*A(1);

mdot0 = mdot;
mdot = rho*u*A(end);
if ht ==1
    Qdot = mdot*(cp*(Tt2-T0*(1+(gam-1)/2*M0^2))-w);
end
```

3.6 rootFind.m

```
classdef rootFind
    %rootFind is a class of functions that find the root of a function /
    %data set
 %-------------------------------------------------------------------------%
    methods (Static)
        function x = Bisect(f,a,b,tol)
            %Bisect uses the bisection algoritm using the interval
            iter = 0;
            while (b-a)/2 >= tol
```

```
                c = (a+b)/2;
                if f(c) > 0
                    b = c;
                end
                if f(c) < 0
                    a = c;
                end
                iter = iter + 1;
            end
            x = (a+b)/2
        end
%-------------------------------------------------------------------------%
        function x = newRap(f,x0)
            %newRap is a function that utilizes the Newton-Raphson
            %algorithm to find the roots of the function
            %x0 is the initial guess
            fp = diff(f);

            x=x0;
            nmax=25;
            eps=1;
            n=0;

            while eps>=1e-5&&n<=nmax
                y=x-double(f(x))/double(fp(x));
                eps=abs(y-x);
                x=y;
                n=n+1;
            end
        end
%-------------------------------------------------------------------------%

    end
end
```

3.7 Tabulated data for finding m0

initial mass [kg]	Max total temp [K]	σ	B	η
54649.57493	601.826366	200	0.8	0.7
41520.31975	478.691559	200	0.8	0.8
30275.62944	422.0662108	200	0.8	0.9
43066.21768	845.901369	200	1	0.7
38709.56885	557.5064317	200	1	0.8
31864.83859	441.7417878	200	1	0.9
32515.892	1356.24459	200	1.2	0.7
33962.39008	695.6193473	200	1.2	0.8
32847.46885	471.3179735	200	1.2	0.9
25052.55585	2468.79628	200	1.4	0.7
28526.47711	938.9907038	200	1.4	0.8
33047.15636	514.9597241	200	1.4	0.9
20609.43319	5035.017163	200	1.6	0.7
23601.65864	1375.871824	200	1.6	0.8
32403.82875	578.828545	200	1.6	0.9
44991.66076	812.4738334	300	0.8	0.7
39697.84885	547.3867717	300	0.8	0.8
32142.30117	439.3608828	300	0.8	0.9
30766.3062	1450.893311	300	1	0.7
32664.27005	718.7549842	300	1	0.8
32369.05699	475.8524933	300	1	0.9
21860.41126	3126.494861	300	1.2	0.7
25258.64394	1062.794396	300	1.2	0.8
31323.57362	534.5263868	300	1.2	0.9
17328.47113	7932.262715	300	1.4	0.7
19376.1976	1775.558293	300	1.4	0.8
29103.01841	627.969451	300	1.4	0.9
15538.88203	23199.8501	300	1.6	0.7
15511.2556	3321.423047	300	1.6	0.8
26123.69347	777.0936353	300	1.6	0.9
37952.32532	1136.450104	400	0.8	0.7
37686.11371	639.2931423	400	0.8	0.8
34127.70066	459.8274624	400	0.8	0.9
24348.96666	2587.776102	400	1	0.7
27831.30225	962.2277593	400	1	0.8
32698.3109	518.7427566	400	1	0.9
17763.9227	7382.833081	400	1.2	0.7
19962.05984	1704.627929	400	1.2	0.8
29604.33285	619.7186666	400	1.2	0.9
15255.87387	25509.77356	400	1.4	0.7
15064.99396	3514.735968	400	1.4	0.8
25536.53446	792.9188825	400	1.4	0.9
14996.27341	104709.1646	400	1.6	0.7
12509.70009	8263.284652	400	1.6	0.8
21420.67832	1094.004166	400	1.6	0.9
33330.82213	1625.445377	500	0.8	0.7
35834.73764	759.9418648	500	0.8	0.8
36262.67803	483.6910647	500	0.8	0.9
21261.60846	4686.729116	500	1	0.7