Final Project

Table of Contents

Alden "Mac" Lamp, Eric Kostoss, Nathan Orsini; 11-25-2019

init

```
clear all
```

I

A major design parameter affecting thermal efficiency of gas turbines is the temperature of the hot gases resulting from the combustion process. Making substantial increase in thermal efficiency relies on increasing the combustion product temperature. However, this temperature is limited by the endurance of turbine components, including the blades. Developing advanced cooling systems is therefore a critical issue that needs to be addressed in the design phase to ensure that the turbine blades can endure high temperature levels. The trailing edge region of the blade is of articular importance as it needs to be kept thin to limit the aerodynamic losses, making it the most vulnerable part of the blade at high temperatures. The state-of-the-art cooling systems for the trailing edge is based on film cooling, consisting of flow of cooler air, coming from blade internal cooling, through slots at the trailing edge, as shown in Fig. 1, to cool down this region.

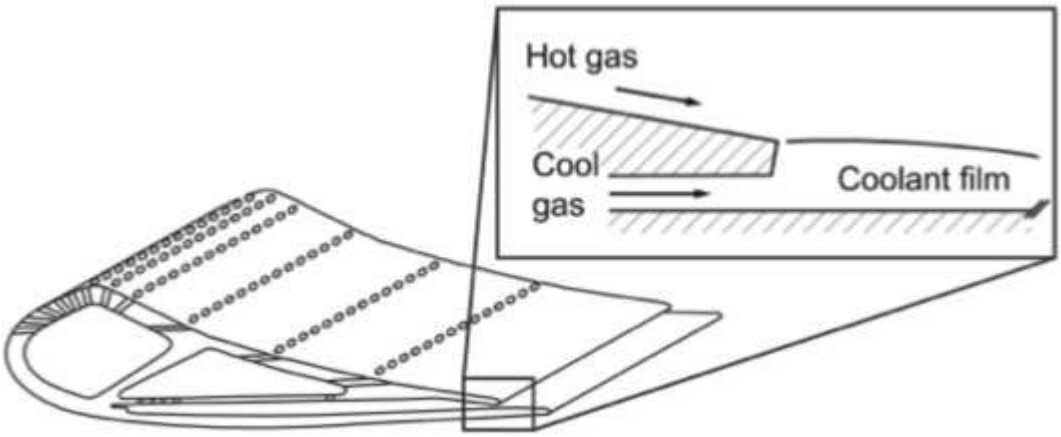
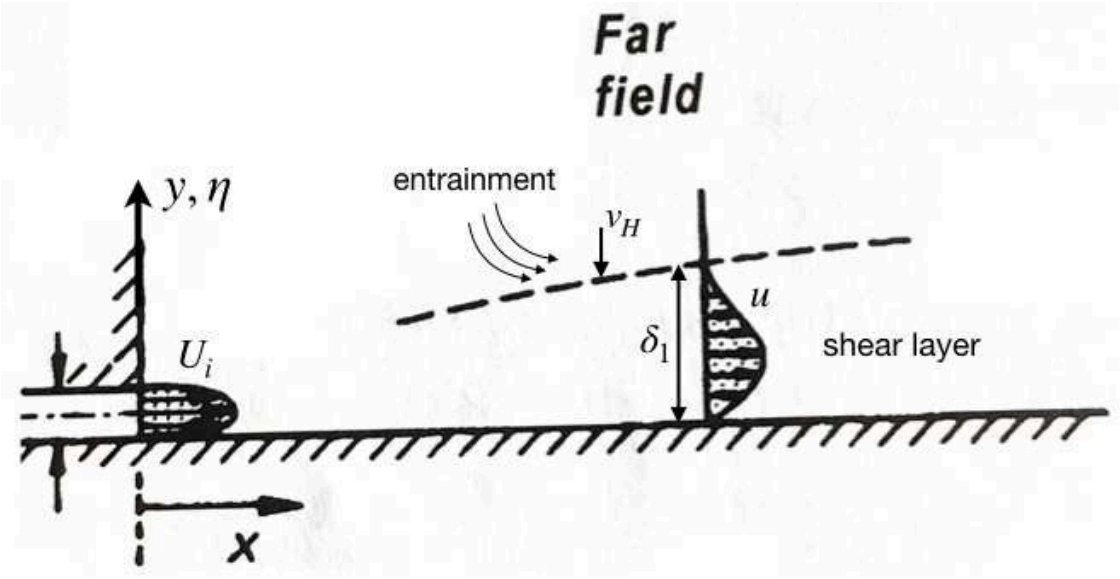


Figure 1: Film cooling of the trailing edge of a gas turbine blade.

The objective of this project is to analyze some of the aerodynamic and heat transfer characteristics of a laminar wall jet which is an idealized flow configuration relevant to film cooling of trailing edge region of gas turbine blades as well as many other applications in, e.g., aerofoil design, combustion chamber wall cooling and air conditioning. Wall jet consists of a jet of fluid emanating from a slot near a solid wall (Fig. 2) and spreading next to the wall as a shear layer.

Consider a laminar wall jet of air with uniform inlet velocity of Ui = 1m/s, the kinematic viscosity is ? = 1.5 × 10?5 m2/s and the density is ? = 1.2kg/m3. The flow has a Reynolds number of which is considered to be smaller than the critical limit and hence, the flow is in the laminar regime; u and v denote the velocity components in the x and y directions, respectively. Such flow is governed by partial differential equations (PDEs) corresponding to incompressible form of Navier-Stokes and the continuity equations. Introducing non dimensional y coordinate, eta=y/x(Re_x)^(1/4), and velocity f'(eta)=df/deta=u/U0, where U0/Ui=4/ Re_x^(½), these PDEs are transformed into the following ordinary differential equation (ODE) which is much easier to solve,

```
f ″′+f f ″+2f ′^2=0
```



```
        Figure 2: Schematic of a laminar wall jet.
```

subject to boundary conditions eta=0 : f=f'=0 eta=H : f'=f''=0 where H refers to far field in y direction (y -> inf). Here, we set H=10.

```
nu=1.5e-5; rho=1.2;%[m^2/s]; [kg/m^3] with laminar valued Reynolds
 Number
Ui=1; Re=@(x) Ui*x/nu;% [m/s]; []
%nondimensional y: eta=y/x*(Re^.25); velocity: df/d'eta = u/Uo
%Uo/Ui = 4/(Re^.5) to ODE  d3f/d'eta^3 + f*d2f/d'eta^2 + 2*(df/
d'eta)^2 = 0
%for eta=0, f=df/d'eta=0;  for eta=H=10, df/d'eta=d2f/d'eta^2 = 0
```

I.a

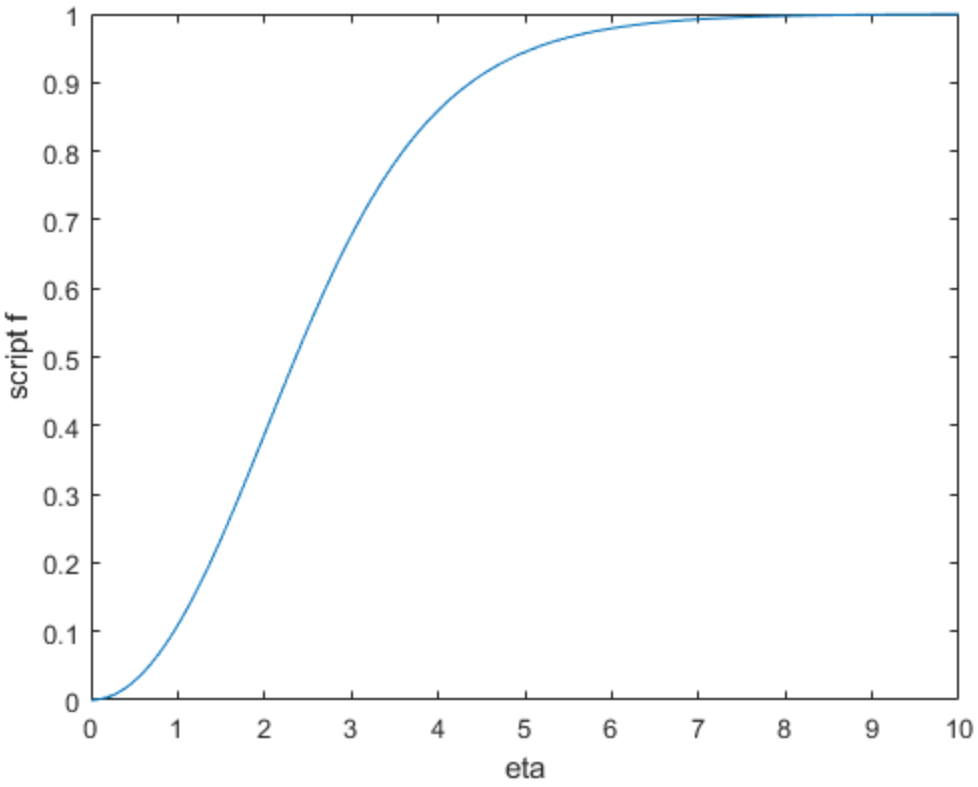
The way that we went about solving this problem as to evaluate the function of Eq 3 along the range of values from 0 to 10 with 0.05 step increments. This can be seen with the etaf function as well as the eta constraint in the code below. After allocating the equation and the constraints, we were able to calculate the f function based off of the values of eta.

```
        implicit solution given as

etaf=@(f,n) log(sqrt(1+sqrt(f)+f)./(1-sqrt(f)))...
    +sqrt(3)*atan(sqrt(3.*f)./(2+sqrt(f)))-n;

        Use this expression to find f(eta) (note that 0<=f<1). Plot f(eta)
        for eta in range of [0, H].

eta=[0:0.05:10]'; f=zeros((10/.05)+1,1);
for i=1:length(eta)
    f(i)=bisectE(@(f) etaf(f,eta(i)),0,1,1e-8);
end
figure(1); hold off; plot(eta,f); xlabel('eta'); ylabel('script f')
```




```
function [root,fx,ARE,trys]=bisectE(fun,xl,xu,ea,tries,varargin)
% bisect: root location zeroes
% [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,p1,p2,...):
% uses bisection method to find the root of func
% input:
% fun= name of function
% xl, xu = lower and upper guesses
% ea = desired relative error (default = 0.0001%)
% tries = maximum allowable iterations (default = 50)
% p1,p2,... = additional parameters used by func
% output:
% root = real root
% fx = function value at root
% ARE = approximate relative error
% trys = number of iterations
if nargin<3, error('bisect(): more arguments please'), end
test=fun(xl,varargin{:})*fun(xu,varargin{:});
if test>0, error('no sign change'), end
if fun(xl,varargin{:})==0, root=xl; ARE=0; go=false;
elseif fun(xu,varargin{:})==0, root=xu; ARE=0; go=false;
else, root=xl; ARE=1; go=true;
end%check the lower and upper points for zero
if (nargin<4)||(isempty(ea)), ea=1e-6; end
if (nargin<5)||(isempty(tries)), tries=50; end
trys=0;
```

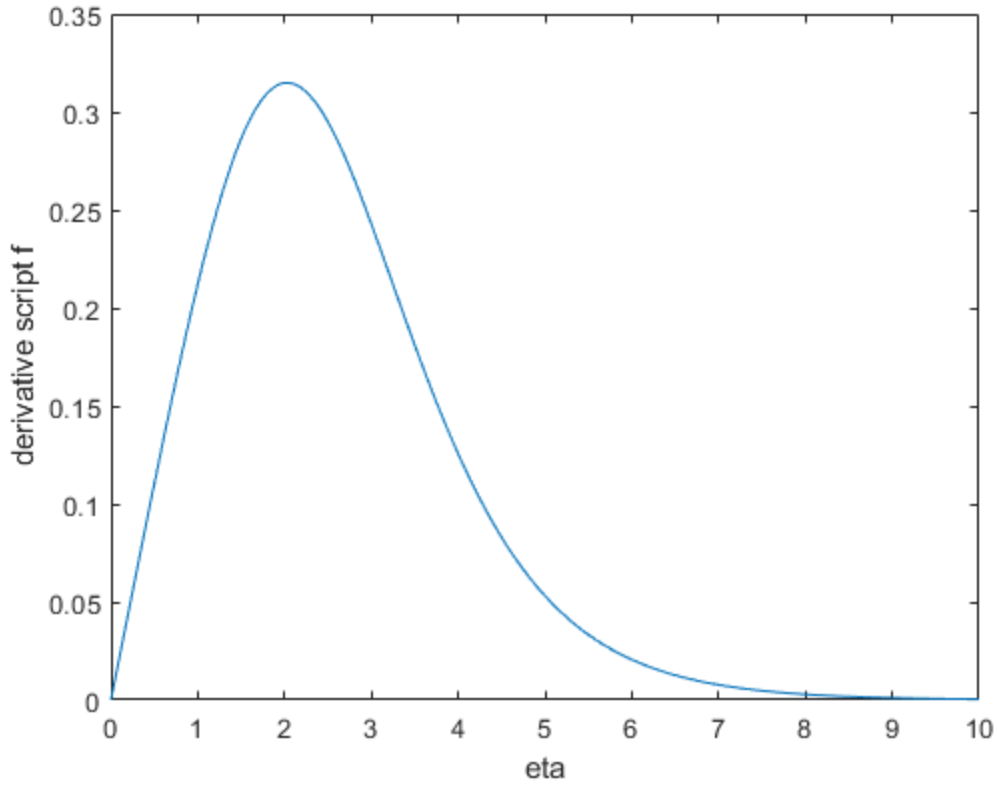
```
while go
  rtold=root;
  root=(xl+xu)/2;
  trys=trys+1;
  if root~=0, ARE=abs((root-rtold)/root); end
  test=fun(xl,varargin{:})*fun(root,varargin{:});
  if test<0
    xu=root;
  elseif test>0
    xl=root;
  else
    if fun(root,varargin{:})==0, ARE=0;
    else, root=xl; ARE=0;
    end%redundantly check lower point for case of upper bound 'Inf'
  end
  if (ARE<=ea)||(trys>=tries), break; end
end
fx = fun(root, varargin{:});
```

I.b

We calculated the derivative of the function f(eta) which we obtained from part A we used the diffc2() function with a step size of 0.05 as what we selected in I.a. The diffc2() function calculates first order derivatives with second order accuracy. This allowed us to get f'(n).

```
        Find f(eta) with at least second order accuracy and plot it vs. eta.

df=diffc2(f,0.05);
figure(2); hold off; plot(eta,df);
xlabel('eta'); ylabel('derivative script f')
```




```
function df=diffc2(f,h)
%%Compute second-order accuracy first derivative
%   input 3+ dependent values at equally spaced intervals, and supply
 the
%   interval and get derivatives for all the points.
l=length(f);
df=zeros(l,1);
if l>2
    df(1)=(-f(3)+4*f(2)-3*f(1))/(2*h);
    for b=[2:l-1]
        df(b)=(f(b+1)-f(b-1))/(2*h);
    end
    df(l)=(3*f(l)-4*f(l-1)+f(l-2))/(2*h);
else
    disp('not enough points for second-order accuracy')
end
```

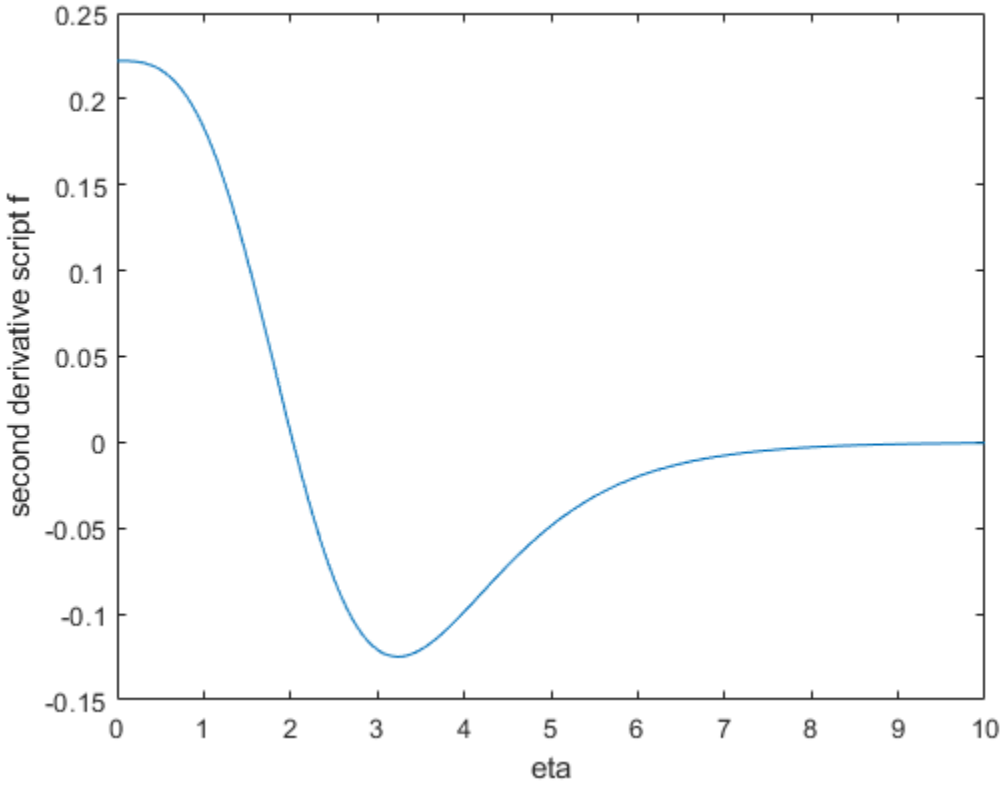
I.c

Similar to part B we used the diffc2() function to calculate the function of f''(eta). we did this using the first derivative that we obtained in Part B and again used the same spacing of 0.05. This allowed us to compare the second order derivative of f(eta) to the skin friction coefficient Cf with some cancellations and rearranging.

```
        Find f''(eta) with at least second order accuracy and plot it vs.
        eta. Show that the skin friction coefficient
        Cf=toaw/(1/2*p*Uinf^2) ~ 1.778/Re_x^5/4 as obtained from theory. toaw is
        the shear stress at the wall toaw=mu*du/dy | y=0

d2f=diffc2(df,0.05);
figure(3); plot(eta,d2f);
xlabel('eta'); ylabel('second derivative script f')
fprintf('Cf*(Re^(5/4))/8: %5.4f, compare %5.4f\n',d2f(1),1.778/8)
```

```
Cf*(Re^(5/4))/8: 0.2222, compare 0.2223
```



I.d

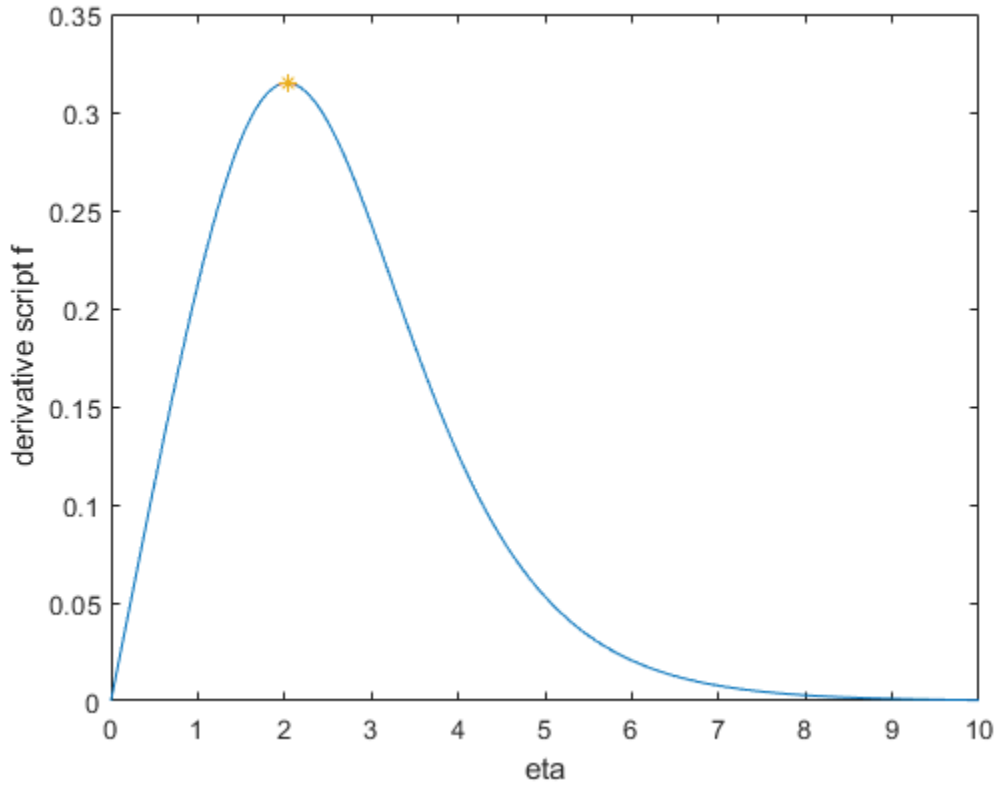
We changed our discrete values of the first derivative into a spline, and wrote an anonymous function for use in both I.d and I.f with parameters for each. Using goldmin() with the spline as an input, and the argument for taking the opposite of the function set to 'true', goldmin() found the value farthest from 0 which we plotted and then compared to expected values given.

```
        Find maximum velocity f'max = umax/Uo value and its eta location.
        Verify that f'max ~ 2^(?5/3) which occurs at eta ~ 2.029 consistent
        with the theoretical values.

dfetaSpln=spline(eta,df);
fetaReg=@(eta,neg,dwn) ((-2*neg)+1)*(ppval(dfetaSpln,eta)-(dwn*.01));
etaMax=goldmin(fetaReg,0,10,1e-6,9999,true,false);
figure(2); hold on;
```

```
plot((etaMax-.1):.01:(etaMax+.1),...
    ppval(dfetaSpln,(etaMax-.1):.01:(etaMax+.1)),...
    etaMax,ppval(dfetaSpln,etaMax),'*')
fprintf('n [eta]: %5.4f, compare 2.029\n',etaMax)
fprintf('f''(n) =  %5.4f, compare %5.4f\n',...
    ppval(dfetaSpln,etaMax),2^(-5/3))
```

```
n [eta]: 2.0287, compare 2.029
f'(n) =  0.3149, compare 0.3150
```




```
function [x,fx,ea,iter]=goldmin(f,xl,xu,es,maxit,varargin)
% goldmin: minimization golden section search
%   [x,fx,ea,iter]=goldmin(f,xl,xu,es,maxit,p1,p2,...):
%     uses golden section search to find the minimum of f
% input:
%   f = name of function
%   xl, xu = lower and upper guesses
%   es = desired relative error (default = 0.0001%)
%   maxit = maximum allowable iterations (default = 50)
%   p1,p2,... = additional parameters used by f
% output:
%   x = location of minimum
%   fx = minimum function value
%   ea = approximate relative error (%)
%   iter = number of iterations
```

```
if nargin<3,error('at least 3 input arguments required'),end
if nargin<4|isempty(es), es=0.0001;end
if nargin<5|isempty(maxit), maxit=50;end
phi=(1+sqrt(5))/2;
d = (phi-1)*(xu - xl); iter = 0;
x1 = xl + d; x2 = xu - d;
f1 = f(x1,varargin{:}); f2 = f(x2,varargin{:});
while(1)
  xint= xu - xl;
  if f1 < f2
    xopt = x1;
    xl = x2; x2 = x1; f2 = f1;
    x1 = xl + (phi-1)*(xu-xl); f1 = f(x1,varargin{:});
  else
    xopt = x2;
    xu = x1; x1 = x2; f1 = f2;
    x2 = xu - (phi-1)*(xu-xl); f2 = f(x2,varargin{:});
  end
  iter=iter+1;
  if xopt~=0, ea = (2 - phi) * abs(xint / xopt) * 100;end
  if ea <= es | iter >= maxit,break,end
end
x=xopt;fx=f(xopt,varargin{:});
```

I.e

The procedure on Part E is focused around the MATLAB fcn trapz(). We used this to calculate the approximate integral of eta with df^2. Then we multiplied by the trapz() of eta, df. This was done to calculate the Z value which is the wall jet momentum flux. After we compared this analytical value to that of the theoretical value in a graph format.

```
        Calculate the wall jet momentum flux and show that it is consistent
        with the theoretical value:128/9*Ui*?^2

Z=trapz(eta,df.^2)*trapz(eta,df);
fprintf('[flux]/(Ui*nu^2*4^3): %5.4f, compare %5.4f\n',Z,128/
(9*(4^3)))
```

*[flux]/(Ui*nu^2*4^3): 0.2220, compare 0.2222*

I.f

Section F reuses the function defined in I.d, with the parameters for a vertical translation by 0.01 set to 'true'. Bisect() finds the zero between the eta value of the maximum of the function and 10, the upper bound of the domain. Compares to 6.72 given.

```
        Find the shear layer thickness delta1 defined as the theta location
        where f'(eta) ~ 0.01. Show that your prediction is in agreement with
        theoretical value delta1 ~ 6.72

delta=bisectE(fetaReg,etaMax,10,1e-8,9999,false,true);
fprintf(strcat('shear layer thickness delta1\n',...
    'delta = %5.4f, compare 6.72\n'),delta)
```

```
shear layer thickness delta1
delta = 6.7397, compare 6.72
```

I.g

The velocity at H was calculated with the us adjusting and simplifying the equation to 3*eta()*df()-f()*Ui = v/Re^(3/4). This was then filled with eta=10, and the values of f and its derivatives also at eta=10, or the last element in the arrays. This allowed for the velocity to be compared to the theoretical value.

```
        Calculate the v velocity at the edge of the layer,
        vH = (3*eta*f'?f)Ui(Rex) ?3/4 at eta = H and verify that your result
        matches the theoretical value vH ? (?1)Ui(Rex) ?3/4. The negative vH
        value indicates that the ambient fluid gets sucked into the shear layer
        as the jet develops downstream (called entrainment)

v_H=(3*eta(end)*df(end)-f(end))*Ui;%eta(end)==10
fprintf('Analytical: %5.4f\n',v_H)

Analytical: -0.9879
```

I.h

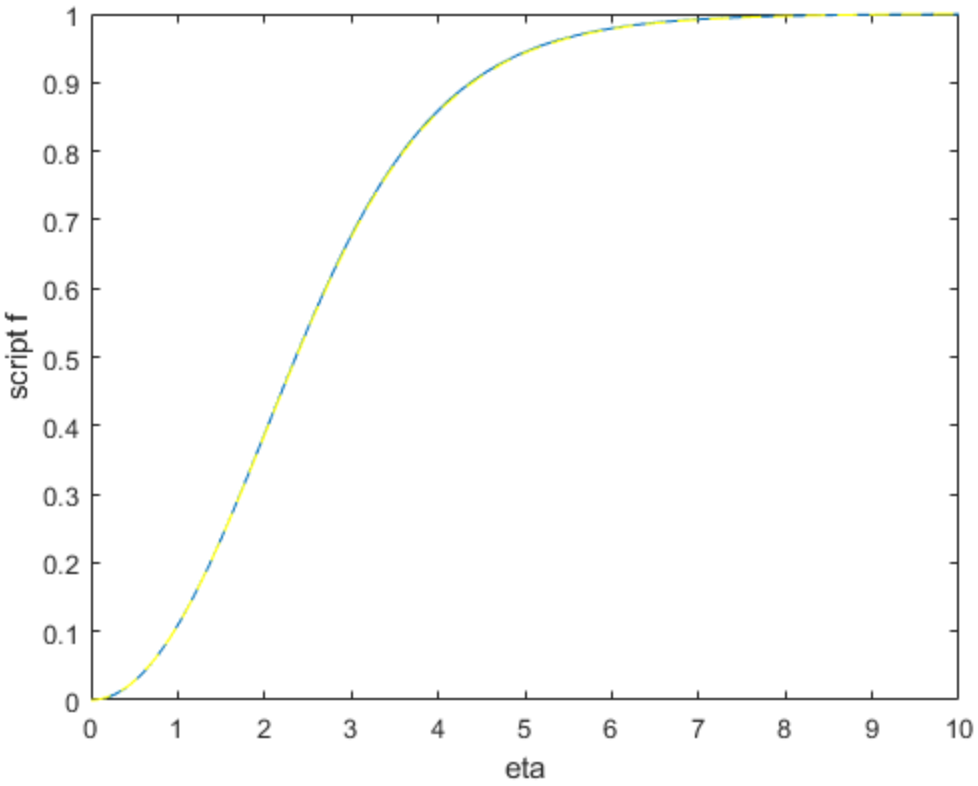
The method for this was to use a built-in Runge-Kutta ODE routine to calculate the value of f(eta) and compared to the solution obtained in Part A. This was done using the MATLAB function ode45. The ode's criteria were based on the length that it was integrating as well as the the t-intercepts of the function. The length of integration was the value of eta, which spanned from 0 to H, with H having the value of 10. The y initial conditions were set to be zero for the first and second derivative, and the last one, the third derivative was solved using the res() fcn to calculate the point by minimizing the values that are inputted and reducing them down to find an answer. The result is plotted on figure(1) with the analytical solution obtained as the inverse of the given solution.

```
        Solve Eq.(1) for f(eta) and compare your solution with that obtained in
        Part (a). Implement the boundary condition at eta=H as f'+f'' = 0.

zah=fzero(@res,0,[],1);
[etaODE,fODE]=ode45(@dydx,[0 10],[0 0 zah],[],1);
figure(1); hold on; plot(etaODE,fODE(:,1),'y--');

Warning: Failure at t=7.211685e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.421085e-14)
at time t.
Warning: Failure at t=6.424893e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.421085e-14)
at time t.
Warning: Failure at t=5.723927e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.421085e-14)
at time t.
```

```
Warning: Failure at t=5.099432e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.421085e-14)
at time t.
Warning: Failure at t=4.543098e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.421085e-14)
at time t.
Warning: Failure at t=4.047437e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.421085e-14)
at time t.
Warning: Failure at t=3.605854e+00.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (7.105427e-15)
at time t.
```



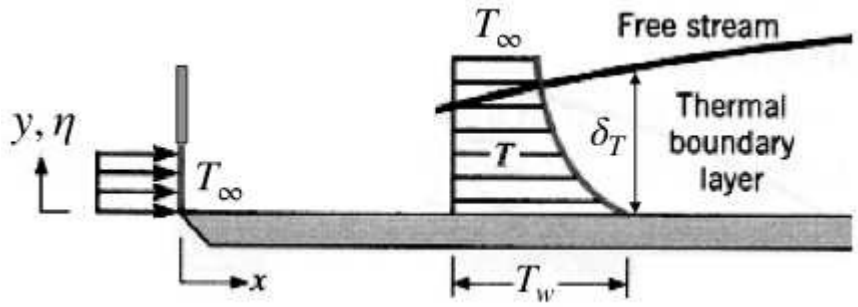
II

The shear layer formed by the wall jet causes variation of temperature near the wall resulting in film cooling of the wall (Fig. 3). The convective heat transfer from the wall is governed by a PDE for temperature

corresponding to energy equation. Using the non-dimensional variables above along with theta(eta) = (T-Tinf)/(Tw-Tinf) we transform this PDE into an ODE for theta

```
d^2theta/deta^2+Pr*f*dtheta/deta=0
```

Subject to boundary conditions eta=0 : theta=1 eta=H : theta=0 where Pr is the Prandtl number, Tw is the temperature of the surface and TInf is the inlet and ambient air temperatures. We set Pr = 0.7




```
Figure 3: Thermal boundary layer over a flat plate.
```

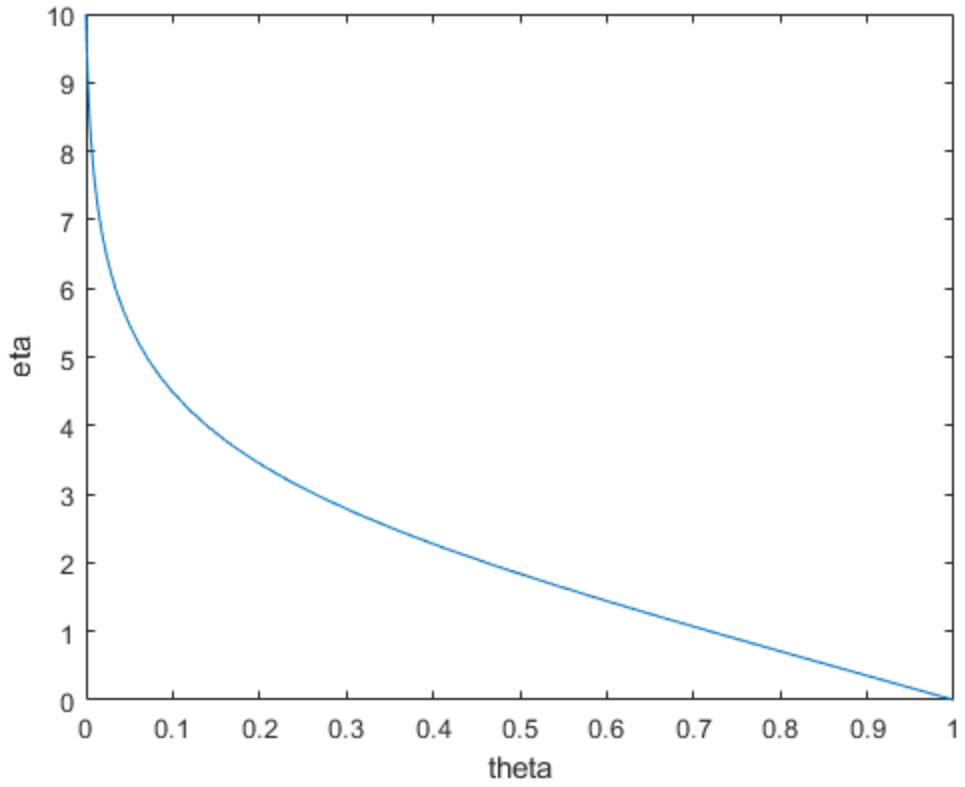
```
Pr=.7;
```

II.i

We calculated the ODE for the problem using ode45() function in MATLAB. To find the value of the end point of the y-intercept, which we named zai, we needed to use the res() function which takes the ODE of dydx function and outputs zai given an initial guess. After calculating for the zai we are able to calculate the ode for eta and theta. This is done using a scale of 0 to 10, and using the initial conditions given which are 0 and zai, these are the values that intersect with the y axis.

```
Solve this ODE to find theta(eta) as a function of eta in the range
[0, H]. Plot theta(eta).
```

```
zai=fzero(@res,0,[],2,fODE(:,1),etaODE);
[n,theta]=ode45(@dydx,[0 10],[1 zai],[],2,fODE(:,1),etaODE);
figure(4); hold off; plot(theta(:,1),n); xlim([0,1]);
xlabel('theta'); ylabel('eta')
%there's a theta value of -1.687e-16 and only one, hence xlim()
```




```
function r=res(za,pick,varargin)%varargin (fn,x)
if pick==1
    [~,y]=ode45(@dydx,[0,10],[0,0,za],[],pick);
    r=y(end,1)-1;
elseif pick==2
    [~,y]=ode45(@dydx,[0 10],[1 za],[],pick,varargin{:});
    r=y(end,1)-0;
end
```

```
function dy=dydx(n,y,pick,varargin)
if pick==1
    dy=[y(2);y(3);-y(1).*y(3)-2.*y(2).^2];%solving for f' f" f"' from
 eq(1)
elseif pick==2
    fnx=spline(varargin{2},varargin{1},n);
    dy=[y(2);-0.7*fnx*y(2)];%solving for dTHETA and dtTHETA from eq(4)
end
```

II.j

The way that we went about solving Part j is that we made a spline using the criteria of eta and theta. Then we moved the spline down by a value of 0.01 this made it so that when we used the bisect function to

find the value of the 0 it would actual be the value of 0.01. This would then display the correct answer that we needed.

```
        Calculate the thickness of the thermal boundary layer deltaT
        (i.e., eta location where theta ~ 0.01) and show that
        deltaT ~ delta1(Pr)^(?1/3) as expected from theory.

thetaEtaSpln=spline(n,theta(:,1));
thEta=@(eta) ppval(thetaEtaSpln,eta)-.01;
deltaT=bisectE(thEta,0,10,1e-8,9999);
fprintf(strcat('thermal boundary layer thickness deltaT\n',...
    'deltaT = %5.4f, compare %5.4f\n'),deltaT,(delta*(Pr^(-1/3))))

thermal boundary layer thickness deltaT
deltaT = 7.5586, compare 7.5906
```

II.k

For 2K we calculated the theta' value and proceeded to perform a central finite difference method on it. This allowed us to calculate for the theta'(0) value which was discovered to be 0.283 which is close to the value of the theoretical that was calculated from given equation. The theoretical value came out to eb 0.2087, which is fairly close to the calculated value that we found.

```
        Calculate the temperature gradient at the wall
        theta'(0) = dtheta/deta|eta=0 with at least second order accuracy and
        show that the predicted Nusselt number compares reasonably well with
        the theoretical value Nu/Re^1/4 = ?theta'(0)~0.235(Pr)^(1/3).

dtheta=diffc2(theta(:,1),n(2)-n(1));
fprintf('theta''(0): %5.4f, compare %5.4f
\n',-1*dtheta(1),.235*(Pr^(1/3)))

theta'(0): 0.2827, compare 0.2087
```

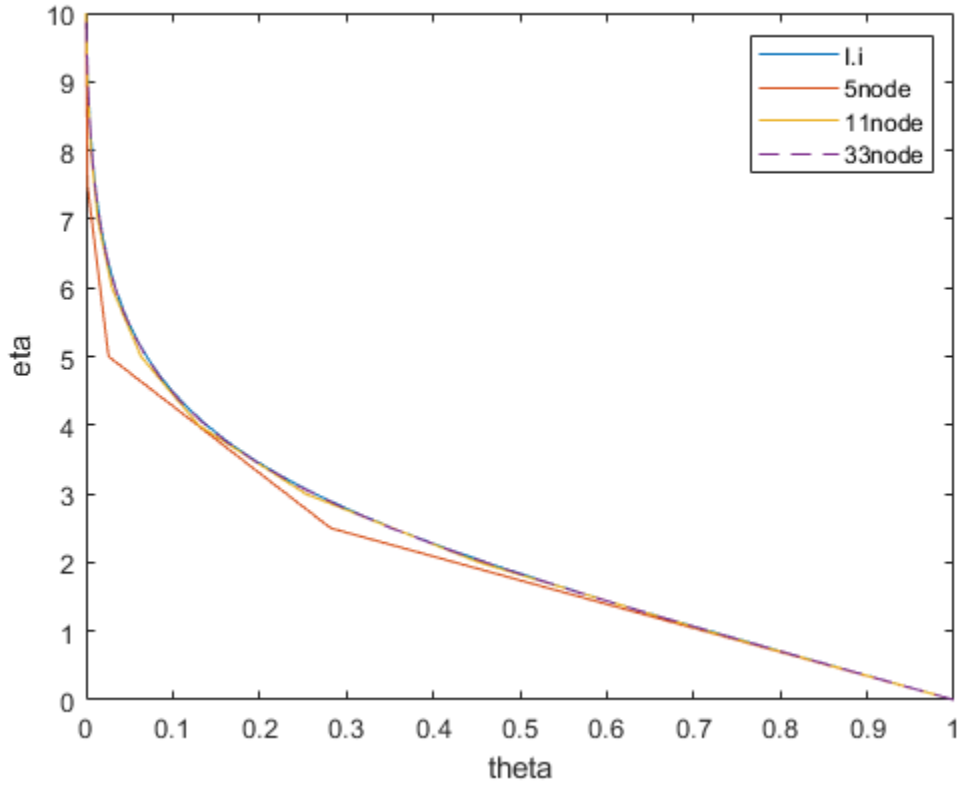
II.l

To solve part L we created a spline using eta and f from section I. Then we use a central finite difference method to solve for theta. The central difference method uses a matrix format to solve for the theta value, the matrix was propagated using a for loop until it reached the proper number of iterations to accurately solve the problem. This was then compared to the graph that was obtained in part I.

```
        Solve Eq. (4) with its boundary conditions using finite-difference
        method. Show verification of your finite-difference solution by
        demonstrating that the solution becomes less sensitive to step size
        change in eta as the step size value decreases (due to decrease in
        truncation error); this can be done by plotting theta versus eta for at
        least three different change in eta values. Compare your solution with
        that obtained in Part (i).

fetaSpln=spline(eta,f);
[x,theta]=CFD(fetaSpln,5);
%f=@(eta) ppval(fetaSpln,eta);
figure(4); hold on; plot(theta,x)
[xx,thetaa]=CFD(fetaSpln,11); [dad,thetaaa]=CFD(fetaSpln,33);
```

```
plot(thetaa,xx,thetaaa,dad,'--');
 legend('I.i','5node','11node','33node')
```




```
function [x,theta]=CFD(func,n)
f=@(eta) ppval(func,eta);
x=linspace(0,10,n);
h=10/(n-1);
A=zeros(n-2,n-2);
Prfx=0.7*f(x)*h/2;
for i=1:n-2-1
    A(i,i)=-2;
    A(i,i+1)=1+Prfx(i+1);
    A(i+1,i)=1-Prfx(i+2);%for next row, use next f
end
A(end,end)=-2;
Prfx=0.7*f(x(2))*h/2;
z=zeros(n-2,1);
z(1)=-(1-Prfx)*1;
%z(end)=-(1+c)*0;
theta=A\z;
theta=[1;theta;0];
```

Published with MATLAB® R2019a