Supplemental File

PREDICTIVE COMBUSTION TRAJECTORY VISUALIZATION MODEL FOR STUDY OF CONVENTIONAL AND ADVANCED DIRECT INJECTION COMPRESSION IGNITION COMBUSTION MODES

by

ALEXANDER HOWARD DeLOACH

The University of Alabama, 2016

This supplemental file contains the important sections of the code used for this thesis research. The main script makes calls to sub-routines in the order presented.

Table of Contents (Complete Code)

Table of Contents (Main Diesel Model - Musculus Mixing + Radial Variation in Phi)

Main Diesel Model - Musculus Mixing + Radial Variation in Phi

```
tic
clear;
```

File and Run Selection

```
%Change these!!!!
filename = 'Conv EGR Sweep Data Input with HighLoadHighSpeed.csv';
numberofruns=9; %there must be manually this many "%f"'s in the ReadHSVariables.m file
runselector =8; %chose based on condition from input file

savefigure=1 %true/false to save figures(0 if not on lab computer)
fontsize=10;% fontsize in plots
```

File creation for figure storage

```
if savefigure==1
   dateandtime=fix(clock);
   date=dateandtime([1 2 3]);
   date_str = strcat('V4_',mat2str(date));
   mkdir(['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str, '\ T_Spray'])
   mkdir(['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str, '\ Full_Screen'])
   mkdir(['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str, '\
Mixing_and_Heating'])
   mkdir(['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str, '\ MFB'])
   mkdir(['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str, '\ PHI'])
   mkdir(['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str, '\
Ignition_Locations'])
end
```

Read and setup variables from csv. file

```
ReadLSVariables;
ReadHSVariables;
InputConfig_v01;



load('FlameTempLookup.mat')

% overwrite default values with those in Run_Parameters
for i=1:size(Run_Parameters)
   eval(Run_Parameters{i});
end
for i=1:size(Inj_Parameters)
   eval(Inj_Parameters{i});
end
for i=1:size(Eng_Parameters)
   eval(Eng_Parameters{i});
end

CASteps = 360/CAres;

P_size = size(Data_Input_Cyl_P);
P_cyl = 100*Data_Input_Cyl_P((P_size(1)-CASteps)/2:3*(P_size(1)-CASteps)/2-1,runselector);
clearvars P_size;
EngineSpeed = EngineSpeedInput(runselector);
AirFlowRate = AirFlowRateInput(runselector);
FuelFlowRate = FuelFlowRateInput(runselector);
IMT = TIVCInput(runselector)+273.15;
EGR = EGRInput(runselector);
ExhO2 = ExhO2Input(runselector)/100;
```

```
ExhCO2 = ExhCO2Input(runselector)/100;
yO2int = yAmbO2*(1-EGR/100)+ExhO2*EGR/100;
MWint = yO2int*32+ExhCO2*EGR/100*44+(1-yO2int-ExhCO2*EGR/100)*28;
xO2int = yO2int*32/MWint;
dtCAstep = (1/(EngineSpeed*6))*CAres; %(1/RPM)min/rot)*(60sec/min)*(rot/360deg)*(CAresdeg/step) = [s/step]
degperms=EngineSpeed*6/1000;
```

Spray Characteristics

```
InjectionRateProfile_v01;
```

Cylinder Characteristics

```
VolAreaMass_v01;
```

Heat Release Rate

```
SimpleHRRwHT_v01;
maxHRRi=find(HRR==max(HRR));
for a=maxHRRi:-1:1 %flattens out Heat Release Rate plot before combustion begins to remove premature combustion
   if HRR(a)==0
      break
   end
end
HRR(1:a)=0;
```

Determine injection and combustion timing in degrees and indices used for timing and plotting

```
[pks, locs]=findpeaks(HRR);
firstHRRpki=locs(1);
firstHRRpkdeg=CrAng(firstHRRpki);
SOCdeg=(SOCi+SOIi)*CAres-180 %Start of Combustion (degree)
EOCdeg=EOCi*CAres-180 %End of Combustion (degree)
SOIdeg=SOIi*CAres-180 %Start of Injection (degree)
IDdeg=SOCdeg-SOIdeg %Ignition Delay (degree)
firstHRRpkdegaSOI=abs(SOIdeg-firstHRRpkdeg); %first peak in HRR (degrees after Start of Injection)
InjDurdeg=InjDur_ms*degperms; %Ignition Duration (degree)
EOIdeg=round(SOIdeg+InjDurdeg) %End of Injection (degree)
EOIi=find(CrAng==12);
%See all mixing
%Run_Time = (max(find(mfbhr>.9,1,'first'),find(BulkTemp>1000,1,'last'))- SOIi)/(EngSpDeg/CAres)
%stop after end of HRR
Run_Time = (EOCi- SOIi)/(EngSpDeg/CAres)
taSOI=0:dtCAstep:Run_Time;
```

```
firstHRRpktaSOI=firstHRRpkdegaSOI/(EngSpDeg); %used in older version to help determine flame structure
```

Plotting HRR and MFB

```
figure(13);
clf %clear figure
%-----set size of save figure(used in all saved figures)------
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];% (inches) [ 0 0 x_length y_length] [origin and upper right corner locations]

%-----Plot Setup-----------
[ax,p1,p2] = plotyy(CrAng,mfbhr,CrAng,HRR,'plot','plot');
axis(ax(1),[-5 55 0 1]) %MFB axis
axis(ax(2),[-5 55 0 2500]) %HRR axis
set(ax(1),'Fontsize',(fontsize-2)); %change MFB axis number fontsize
set(ax(2),'Fontsize',(fontsize-2)); %change HRR axis number fontsize
%-----setting axis label and position (done like this to move axis title to fit in figure)(used in most plots)-------
ylabel(ax(1),'MFB','Color','k','Fontsize',fontsize) % label left y-axis
xxxx=ylabel(ax(2),  'Heat Release Rate (J/s)','Color','k','Fontsize',fontsize);% label right y-axis
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1)+2 P(2) P(3)]) %(x, y, z)
xxxx=xlabel('Crank Angle (deg aTDC)','Fontsize',fontsize); % label x-axis as mentioned above
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1) P(2)-.0275 P(3)])
%-----End setting axis label and position---
set(p1,'Color','k','LineWidth',3)
set(p2,'LineStyle','--','LineWidth',3,'Color','r')
set(ax(1),'YColor','k','YTick', 0:.20:1);%changes MFB gridline locations
box(ax(1),'off')
set(ax(2),'YColor','k','YTick', 0:250:2500);%changes HRR gridline locations
grid on
%-----scaling to fit everything in figure(Used in all figures)-------
scalefactor=.9; %just change this value
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   fname0 = sprintf('MFB&HR_%i.png',runselector);
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str  '\'
fname0])
   %(dpng-save a png file, -r600 -resolution per inch)
end
```

Plotting Cylinder Pressure and Bulk Gas Temperature (see Figure 13 code for explanation)

```
figure(14);
clf
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];%(inches)
[ax,p1,p2] = plotyy(CrAng,P_cyl,CrAng,BulkTemp,'plot','plot');
axis(ax(1),[-5 55 0 15000]) % Cylinder Pressure axis
axis(ax(2),[-5 55 0 2000])  % Bulk Gas Temperature axis
set(ax(1),'Fontsize',(fontsize-2));
set(ax(2),'Fontsize',(fontsize-2));
ylabel(ax(1),'Cylinder Pressure (kPa)','Color','k','Fontsize',fontsize) % label Cylinder Pressure y-axis
xxxx=ylabel(ax(2),'Bulk Gas Temperature (K)','Fontsize',fontsize);% label Bulk Gas Temperature y-axis
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1)+2 P(2) P(3)])
xxxx=xlabel('Crank Angle (deg aTDC)','Fontsize',fontsize); % label x-axis
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1) P(2)-200 P(3)])
set(p1,'Color','k','LineWidth',3)
set(p2,'LineStyle','--','LineWidth',3,'Color','r')
set(ax(1),'YColor','k','YTick', 0:1500:15000);%changes Cylinder Pressure gridline locations
box(ax(1),'off')
set(ax(2),'YColor','k','YTick', 0:200:2000);%changes Bulk Gas Temperature gridline locations
grid on
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   fname0 = sprintf('P_cyl&Bulk_Temp%i.png',runselector);
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str  '\'
fname0])
   %(dpng-save a png file, -r600 -resolution per inch)
end
```

Run Musculus Mixing model version

```
MusculusMixing_v05_4_Alex;
```

Plotting final ignition locations

```
figure(110)
clf
```

```
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];
contour3(Xedges,Yedges,mtotal_current,linspace(0,max(max(mtotal_current)),bins),'fill', 'on')
colormap(jet)
view(0,90)
xxxx=xlabel('Local Reaction Temperature (K)','FontSize',fontsize);
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1) P(2)-0.1 P(3)])
ylabel('Equivalence Ratio (air based)','FontSize',fontsize);
grid off
set(gca,'Fontsize',(fontsize-2))
colorbar
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str '\
Ignition_Locations-final frame']);
end
%(dpng-save a png file, -r600 -resolution per inch)
```

Plotting final post mixing and heating

```
figure(111)
clf
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];
contour3(Xedges,Yedges,mtotal_current2,linspace(0,max(max(mtotal_current2)),bins),'fill', 'on')
colormap(jet)
view(0,90)
xxxx=xlabel('Local Reaction Temperature (K)','FontSize',fontsize);
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1) P(2)-0.1 P(3)])
ylabel('Equivalence Ratio (air based)','FontSize',fontsize);
set(gca,'Fontsize',(fontsize-2))
grid off
colorbar
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
```

```
    print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\' date_str '\
Heating_and_Mixing-final frame']);
    %(dpng-save a png file, -r600 -resolution per inch)
end
```

Final Plots

```
figno=5;
figure(figno);
plotvariable = 'HRR';
plotCAlow=-100;
plotCAhigh=100;
plotilow=SOIi;
plotihigh=(plotCAhigh+180)/360*CASteps;

eval(['plot(CrAng(plotilow:plotihigh),' plotvariable '(plotilow:plotihigh))']);
eval(['axis([plotCAlow plotCAhigh min(' plotvariable ')-.05*max(' plotvariable ') 1.05*max(' plotvariable ')])']);



toc
```

Published with MATLAB® R2015a

InjectionRateProfile_v01

```
%generates injection profile (kg/s) after start of injection


SOI = MainInjectionSOIInput(runselector)+SOImod/1000*EngineSpeed*6;
SOI = SOI - mod(SOI,CAres);
CAStepsASOI = round((180-SOI)/CAres);
SOIi = CASteps-CAStepsASOI;

InjDur_ms = MainInjectionDurationInput(runselector)/degperms*InjDurmod;
InjVelocity(2,:)=[0 0.07e-3 .08e-3 1e-3*(InjDur_ms-.1) 1e-3*InjDur_ms 1];


mfinj_tot=zeros(CAStepsASOI,1);
Unoz=zeros(CAStepsASOI+1,1);
mdotf_noz=zeros(CAStepsASOI+1,1);
tasoi=zeros(CAStepsASOI+1,1);


check=1;
while(check)
for(i=1:CAStepsASOI)
   Unoz(i) = InjVel_v01(InjType,InjVelocity,i*dtCAstep);
   mdotf_noz(i) = Unoz(i)*rho_f*pi/4*InjNozDiameter^2*Ca;
   tasoi(i) = dtCAstep*i;
   if(i>1) mfinj_tot(i) = mfinj_tot(i-1)+mdotf_noz(i)*dtCAstep; end
end

Calcfuelflow = mfinj_tot(CAStepsASOI)*EngineSpeed/120*1000*NumberofHoles; %g/s/cylinder
%Check Fuel Flowrate calculated versus measured

DiffFuel = abs(FuelFlowRate-Calcfuelflow)/FuelFlowRate*100;
if(DiffFuel<1)
   check=0;
else
   InjVelocity(1,:) = InjVelocity(1,:)*FuelFlowRate/Calcfuelflow;
end
end

clearvars check;
```

VolAreaMass_v01

```
CrAng = zeros(CASteps,1);
dPdxta = zeros(CASteps,1);
Volume = zeros(CASteps,1);
dVdxta = zeros(CASteps,1);
Area = zeros(CASteps,1);

Vc = pi*Stroke*Bore^2/4/(CR-1);
dxta = 2*pi/CASteps;

for(i=1:CASteps)
CrAng(i) = -180+(i-1)*CAres;
xta = i*dxta-pi;
A=Stroke/2;
X=(ConRod+A)-A*cos(xta)-sqrt(ConRod^2-A*A*sin(xta)^2);
Volume(i)=Vc+pi*Bore*Bore/4*X;
Area(i)=pi*Bore*(X+Bore/2.0);
if(i==1)
   dPdxta(i)=0;
   dVdxta(i)=0;
else
   dPdxta(i) = (P_cyl(i)-P_cyl(i-1));
   dVdxta(i) = (Volume(i)-Volume(i-1));
end
end

iivc = cast((DegIVC+180)/CASteps,'int16')+1;

Airmass = Volume(iivc)*P_cyl(iivc)/Ramb/IMT;



Airmass = AirFlowRate/(1-EGR/100)/EngineSpeed*120/1000;
IMT = Volume(iivc)*P_cyl(iivc)/Ramb/Airmass;




IMTc = IMT - 273.15

BulkTemp = P_cyl.*Volume/Airmass/Ramb;
rho = P_cyl/Ramb./BulkTemp;
clearvars check xta A X IMTc;
```

SimpleHRRwHT_v01

```
%NEED TO DO:
%1. Make if statment on using gamma based on SOC
%2. Make constant gamma a function of ambient mixture (frozen based on
%intake species.



EngSpDeg = EngineSpeed*360/60; %[deg/s]
EngSpStep = EngSpDeg/CAres; %[steps/s]
TotBurnedFuel = mfinj_tot(CAStepsASOI)*NumberofHoles*CombustionEfficiencyInput(runselector);

%Constant leading parameters of Hohengberg HT Calculations
hohparm = .00326*(EngineSpeed*Stroke/30+1.4)^.8; %0.00326 - J/s to kJ/s constant
HRR = zeros(CASteps,1);
Qnet = zeros(CASteps,1);
mfbhr = zeros(CASteps,1);
Hoh = zeros(CASteps,1);
HT = zeros(CASteps,1);
MaxHRR = 10000;
i=-1; j=-1;
Burnedfuel=0;
check=false;
notdone=true;

SOCi=0;
SOCcheck=false;



while(notdone&&j<100)
   j=j+1;
   while(i<(CAStepsASOI-1))
      i=i+1;

         Hoh(i+SOIi)=Volume(i+SOIi)^(-.06)*P_cyl(i+SOIi)^(.8)*BulkTemp(i+SOIi)^(-.4)*hohparm;
         HT(i+SOIi)=Abal*Hoh(i+SOIi)*Area(i+SOIi)*(BulkTemp(i+SOIi)-T_wall); %[kJ/s]

         if(i>0)
            if((Qnet(i+SOIi-1)>50)||check)
               gamma=1.35-6E-5*BulkTemp(i+SOIi)+1.08e-8*BulkTemp(i+SOIi)^2;
               check=true;
            else
               gamma = 1.4;
            end
         else
            gamma=1.4;
```

```
        end

        Qnetrad = Abal*(gamma/(gamma-1)*P_cyl(i+SOIi)*dVdxta(i+SOIi)+1/(gamma-1)*Volume(i+SOIi)*dPdxta(i+SOIi)); %[kJ/step]
        Qnet(i+SOIi)=Qnetrad*EngSpStep; %[kJ/s]

        if(i>(find(dPdxta/max(dPdxta)>EarlyInjectionFalseHRRemovePercent,1,'first')-SOIi))
            HRR(i+SOIi) = Qnet(i+SOIi)+HT(i+SOIi);
            if(HRR(i+SOIi)>0)
                Burnedfuel = HRR(i+SOIi)/EngSpStep/Qlhv + Burnedfuel;
                if(SOCcheck==false)
                    SOCi=i;
                    SOCcheck=true;
                end
            else
                HRR(i+SOIi)=0;
            end

            mfbhr(i+SOIi)= Burnedfuel/TotBurnedFuel;
        else
             HRR(i+SOIi)=0;
             mfbhr(i+SOIi)=mfbhr(i+SOIi-1);
        end


end


[maxHR, i] = max(HRR);

while(HRR(i)>HRRCutoff*maxHR&&i<(360/CAres))
    i=i+1;
end

EOCi=i;

HRR(EOCi:end)=0;
mfbhr(EOCi+1:end)=mfbhr(EOCi);


if(mfbhr(EOCi)<=1 && mfbhr(EOCi)>0.97)
    notdone=false;
elseif(mfbhr(EOCi) >= 1)
    Abal = Abal - .1;
    i=-1;
    Burnedfuel=0;
else
    Abal = Abal + .01;
```

```
        i=-1;
        Burnedfuel=0;
    end

end

clearvars check TotBurnedFuel i notdone checkloc...
    hohparm Burnedfuel gamma...
    Qnetrad HRCutoff;
```

Published with MATLAB® R2015a

Table of Contents (MusculusMixing_v05_4_Alex)

MusculusMixing_v05_4_Alex

```
%This version removes the plotting but retains the data export file.
%
%Added varaible density that changes the ambient based on cylinder
%pressure. This creates slighlty lower penetration distances and higher
```

```
%centerline equivalance ratio holding other things equal.
%
%
%TODO: Figure out why time step is so dependent on the mass accumulation
%code... it gets unstable with dt above 5e-7
```

Initializations and Model Setup

```
radialsteps=10; %Change radial CV resolution

plotnumber=0;
counter=0;
%Used to pause code at specific MFB points for capturing plots
test1=1;
test2=1;
test3=1;

%---Initialized spray matrixes
FlameHist = [];
PhiHist = [];
mfbstepcolumn=[];
FlameHist2 = [];
PhiHist2 = [];
mtotal_cumulative_column=[];



bins=200; %Change phi-T plot resolution
phileanedge=0.8;
mtotal_current=zeros(bins,bins);
mtotal_current2=zeros(bins,bins);
imovie1=0;
```

```
%**************************************************************************%
%  Program Title: 1-Dimensional Transient Diesel Jet Model               %
%  Created By: Kyle Kattke                                               %
%  Created On: 6/4/07 - 8/23/07                                          %
%  Modified by: MPM, LMP 6/11                                            %
%               Alexander DeLoach 10/17/2016                             %
%                                                                        %
%  Function: Program is designed to model the injection of fuel from a   %
%            fuel injector into a non-combusting, still environment.  It %
%            predicts velocity and equivalence ratio within the jet,     %
%            as well as penetration with time and entrainment rate       %
%                                                                        %
%  Domain: The entire spray domain is discretized into small CVs.  They  %
%          are initialized to equal widths.                              %
%                                                                        %
```

```
%  Method: First, the mass and momentum in the CVs are calculated from    %
%          the fluxes for the previous time step.  The new mean          %
%          velocities are calculated, from which the flows in the CVs at  %
%          the current time step are calculated.                         %
%                                                                        %
%          Each given time step is calculated based on a set maximum     %
%          Courant number.                                               %
%                    !                                                   %
% Comments: User inputed values are all listed in top portion of code.    %
%          There are many toggles to turn various features on or off.    %
%                                                                        %
%          Many equations and comparisons were made using methods and    %
%          equations listed in the paper 'Effects of Gas Density and     %
%          Vaporization on Penetration and Dispersion of Diesel Sprays'  %
%          by Naber and Siebers.                                         %
%                                                                        %
%  Children Functions: (1) - Contour_Plotter.m                           %
%                                                                        (2) - InjVel.m
%                                                                        (3) - LLFA.m
%************************************************************************%
tic %timer
```

Default User Inputs

```
% Some or all of these values may be overwritten using the variable
% 'Run_Parameters', which contains a list of assignment commands, executed
% after all of the default inputs are set (see below)
rho_a = rho(SOIi);          % [kg/m^3] Ambient density
T_a = BulkTemp(SOIi);  % [K] ambient temperature (for liquid length calc only
BT = 0.5;              % [-] F/A ratio by mass for complete vaporization of liquid
fontsize=10;           % fontsize in plots
linewidth = 2;
a_Siebers = 0.683;     % [-] spreading angle correction factor for Siebers
% 0.683 gives approx same penetration with uniform velocity profile
% as Abramovich's parabolic velocity profile (real jet)
total_length = 0.125;   % [meters] Domain size
dx = 1e-3;             % [meters] CV widths
Courant = .2;          % [-] Courant number
velocity_profile = 1;  % [-] set to 0 for uniform velocity and 1 for Abramovich (real jet)
adjust_theta = 1;      % [-] flag to alter theta uniform velocity profile
dtmax=dtCAstep;            % [s] time step
% interval between image updates

M_threshold = 0.05;    % Threshold for measuring penetration by Momentum flux (vapor head)
phi_threshold = 0;     % Penetration threshold by equivalence ratio (liquid length)
% set to zero to use Siebers' method to determine equivalence ratio
% at the liquid length
```

```
do_plot=1;



outputupdatestep=100;
plotphi=1;

clearvars dout;
```

Kinetics Interpolation Data from CANTERA

```
PhiInterp=[0 0.5 0.75 0.9 1 1.125 1.25 1.375 1.4375 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 3.75 4 4.25 4.5 4.75 5 6 10000];
percentInterp=[0 0.02250962 0.194800408 0.414713574 0.591124786 0.804265105 0.952825683 1 0.985334797 0.95079994 0.727435382
0.510991156 0.351463878 0.242182695 0.169034417 0.120031103 0.086667889 0.063474883 0.047064219 0.035314515 0.026829184
0.02066137 0.01614414 0.012800754 0 0];
```

Set Color Contour Levels

```
Phi_Contour_levels=[.2 .4 .6 .8 1 2 4 6 8 10];
Flame_Contour_levels=linspace(800,roundn(interp2(FlameLookup.EGR,FlameLookup.Phi,FlameLookup.Temp,EGR,1),2),10);
```

Initializations

```
clear Md mdf AreaContract ujet;
D_f = InjNozDiameter*sqrt(Ca);              % Adjusted nozzle diameter
A0 = pi*D_f^2/4;                    % [m^2] adjusted injector cross sectional area
if velocity_profile == 0 && adjust_theta == 1
   InjSpreadAngle=2*atand(tand(InjSpreadAngle/2)*a_Siebers); % [degrees] Siebers angle for uniform velocity calc.
end
x0=D_f/2/tand(InjSpreadAngle/2);              % [m] theoretical jet point origin

% set phi threshold for liquid length based on Siebers' method, using look-up table
if (phi_threshold == 0)       % calculate the phi threshold to give n-C17 liquid length
   FA = LLFA(T_a,rho_a,'c17');      % F/A ratio at liquid length for given conditions
   phi_threshold = AFs*FA;  % equivalence ratio at liquid length
   if velocity_profile==0
      phi_threshold = phi_threshold*.25/.41; % adjustment for "b" in 1999-01-0528
   end
end
phirichedge=phi_threshold;
% Set up domain
% x represents the coordinates of the control volume boundaries, so x is one element greater
% than the total number of control volumes
x = [dx/2:dx:total_length];       % coordinates of CV centers, relative to nozzle exit
A = pi*(tand(InjSpreadAngle/2)*(x+x0)).^2;  % area at CV centers
V = A*dx;                     % CV Volumes (treat as cylinders)
```

```
xsquare=x;
for i=1:radialsteps-1
   xsquare=[xsquare;x];
end

% calculate alpha, beta, and gamma of the velocity distribution
alpha = ones(size(A))*1.5;   % Initialize alpha with fully-developed profile
i=1;
while(i)                  % previous alpha not yet steady
   r = rho_f/rho_a;         % density ratio
   P = [6*(A(i)/A0-1) -(7+18*r) -(2+33*r) -20*r -4*r]; % 4th order polynomial coefficients
   alpha(i) = max(roots(P));    % largest polynomial root is alpha
   if alpha(i)<1.5
      alpha(i) = 1.5;          % fully developed jet reached, so end loop
      i=0;
   else
      i=i+1;                % fully developed jet not yet reached, so continue loop
   end
end

beta = 6*(alpha+1).*(alpha+2)./(3*alpha+2)./(2*alpha+1); % integrated velocity profile factor
gamma = (alpha+1).*(alpha+2)./alpha.^2; % for converting from centerline to average velocity
if velocity_profile==0  % if uniform velocity profile is used
   beta = ones(size(beta));
   gamma = ones(size(gamma));
   alpha = ones(size(alpha))*1e99;
end
```

Calculate steady jet quantities

```
mdotf0 = rho_f*A0*max(InjVelocity(1,:));  % fuel mass flux at nozzle, max in injection rate profile chosen as "steady"
Mdot0 = rho_f*A0*max(InjVelocity(1,:))^2; % Momentum flux at nozzle, max in injection rate profile chosen as "steady"
Ubarsteady = ((rho_a-rho_f)/rho_f*mdotf0+sqrt(((rho_f-rho_a)/rho_f*mdotf0)^2 ...
   +4*beta*rho_a.*A*Mdot0))/2./beta/rho_a./A; % Averaged velocity
Xfsteady = mdotf0./beta/rho_f./Ubarsteady./A; % fuel volume fraction in center of CVs
mfsteady = rho_f*Xfsteady.*A*dx;              % fuel mass flux for steady jet
dbetadx = (beta-[1 beta(1:end-1)])/dx;        % beta gradient
EntRatesteady = rho_a*Mdot0*(2*A./(x+x0)+A.*dbetadx./beta)...
   ./sqrt(((rho_f-rho_a)/rho_f*mdotf0)^2+4*beta*rho_a.*A*Mdot0) ...
   +1/2*((rho_a-rho_f)/rho_f*mdotf0+sqrt(((rho_f-rho_a)/rho_f*mdotf0)^2 ...
   +4*beta*rho_a.*A*Mdot0))*(-1)./beta.^2.*dbetadx; % Entrainment rate
RelEntRatesteady = EntRatesteady*dx./mfsteady;          % relative entrainment rate
phisteady = rho_f*Xfsteady./(rho_a.*(1-Xfsteady))*AFs; % steady jet equivalence ratio
FAsteady = rho_f*Xfsteady./(rho_a.*(1-Xfsteady));
% calculate steady-jet penetration
M = cumtrapz([0 x],[0 (rho_a+(rho_f-rho_a)*Xfsteady).*A.*Ubarsteady]); % integrated momentum
t_steady = M/Mdot0;  % time to deliver integrated momentum at maximum nozzle Mdot rate
```

```
t = 0;                          % [s] run time in while loop
mft = zeros(size(x));           % Initial CV fuel mass total
mfb = zeros(size(x));           % Initial CV fuel mass burned
M =zeros(size(x));              % Initial CV momentum
m = V*rho_a;                    % Initial CV mass (filled with ambient gas)
Ubar = zeros(size(x));          % Initial CV velocity
Xf = zeros(size(x));            % Initial fuel mass fraction
rho_x = rho_a*ones(size(x));    % Initial density of air at SOI
Unoz = InjVel_v01(InjType,InjVelocity,0); % velocity at nozzle
i = 1;                          % i = 1 refers to time = 0*dt
%Penetration1=0;                % [m] based on momentum flux (jet head)
%Penetration2=0;                % [m] based on equivalence ratio (liquid length)
imovie = 0;                     % used in movie loop
mfbstep = 0;
```

Initialize Matrices

```
Arraysizetime=cast(InjDur_ms/1000/(dx/max(InjVelocity(1,:))*Courant)+ (Run_Time-InjDur_ms/1000)/dtmax,'uint16');
AreaContract = zeros(Arraysizetime,1);
Md = zeros(Arraysizetime,1);
mdf = zeros(Arraysizetime,1);
ujet = zeros(Arraysizetime,1);
Penetration1 = zeros(Arraysizetime,1);
Penetration2 = zeros(Arraysizetime,1);
mfbi=zeros(Arraysizetime,1);


phiradvar=zeros(radialsteps,numel(x));
radial=zeros(radialsteps,numel(x));
axial=zeros(radialsteps,numel(x));
mfuradvar=zeros(radialsteps,numel(x));
mftradvar=zeros(radialsteps,numel(x));
FlameTradvar=zeros(radialsteps,numel(x));
percentHR=zeros(radialsteps,numel(x));
phiradvar3=[];
```

Start of Main Iteration

Spray Calculations

```
while t(end) < Run_Time

   mdotf0 = A0*rho_f*Unoz;          % fuel mass flux at nozzle
   Mdot0  = A0*rho_f*Unoz^2;                    % momentum flux at nozzle

   AreaContract(i) = Ca;
   Md(i) = Mdot0;
```

```
mdf(i) = mdotf0;
ujet(i) = Unoz;



mdotf = beta.*A*rho_f.*Xf.*Ubar;  % fuel mass flux in jet
Mdot = beta.*A.*rho_x.*Ubar.^2;     % momentum flux in jet

%Calculate time step based on Courant number and U values
dt=dtmax; % limit time step size for situations where Ubar is low (like start of ramping injection)
Umax = max([Unoz Ubar 1]);
dt = min([dt dx/Umax*Courant]);
% Calculate new fuel mass and momentum in CVs




mft = mft + ([mdotf0 mdotf(1:end-1)] - mdotf)*dt;
M  = M  + ([Mdot0  Mdot(1:end-1)] -  Mdot)*dt;


delta_mdotfb=[0 mdotf(1:end-1)] - mdotf;
test=sum(mfb);
mfb = mfb + delta_mdotfb*dt.*[0 mfb(1:end-1)]./[0 mft(1:end-1)];
mfb(isnan(mfb))=0;

if(sum(mfb)~=0)
   mfb = mfb*test/sum(mfb);
end




t=[t t(end)+dt];   % update time
i=i+1;            % update iteration count


% Take care of situation where volume of fuel exceeds volume of CV
% (can occur near injector for some ramped injections)
% Put extra fuel mass and momentum into next CV downstream
Vf = mft/rho_f;          % volume of fuel in each CV
ii = find(Vf>V);
zzz=0;
while ~isempty(ii)
   zzz=zzz+1;
   jj=ii(1);
   mass_swap =  rho_f * (Vf(jj)-V(jj))*1.001; % 1.001 keeps phi < infinity
```

```
    mft(jj+1) = mft(jj+1) + mass_swap;
    M(jj+1) = M(jj+1) + mass_swap/mft(jj) * M(jj);
    M(jj) = M(jj) - mass_swap/mft(jj) * M(jj);
    mft(jj) = mft(jj) - mass_swap;
    Vf = mft/rho_f;
    ii = find(Vf>V);
end

zz(i)=zzz;

ii = find(mfb>mft);
while ~isempty(ii)
    jj=ii(1);
    mass_swap =  mfb(jj)-mft(jj);
    mfb(jj+1:VLi) = mfb(jj+1:VLi) + mass_swap/(VLi-jj+1);
    mfb(jj) = mft(jj);
    ii = find(mfb>mft);
end



% Calculate new CV fuel and ambient gas quantities
Xf = Vf./V;             % volume fraction of fuel
Va = V-Vf;                      % volume of ambient gas in each CV

ma = Va*rho(SOIi+cast(t(end)*EngSpStep,'int16'));          % mass of ambient gas in each CV
m  = mft+ma;             % total mass in CV
rho_x = m./V;             % CV densities
FA_Ratio =[mft./ma];      % fuel-to-ambient ratio
%------------------------New phi calculations--------------------------
V_dot_air=AirFlowRate/(rho_a*1000);  %Volume flow rate of Fresh Air in,mdotair/densityair %m^3/sec/cyl
m_dot_EGR=AirFlowRate*(1/(1-(EGR/100))-1); %mass flow rate of EGR air in
V_dot_EGR=m_dot_EGR/(rho_a*1000); %Volume flow rate of EGR air in
V_dot_total=V_dot_air+V_dot_EGR; %Tolal Volume flow rate air in
V_dot_O2=V_dot_air*.21+V_dot_EGR*ExhO2; %Volume flow rate O2 in
x_dot_O2=V_dot_O2/V_dot_total; %Percent 02 per total volume air in
phi = FA_Ratio.*AFs/(x_dot_O2/.21);

% Calculate average velocities from CV mass and momentum
Ubar = M./m;                            % mean velocity in jet
Unoz = InjVel_v01(InjType,InjVelocity,t(end));     % velocity at nozzle


HRRhighres = interp1(taSOI,HRR(SOIi:numel(taSOI)+SOIi-1),t(end)); %Calculate HRR rate at each time step for plotting purposes
mfbhrhighres = interp1(taSOI,mfbhr(SOIi:numel(taSOI)+SOIi-1),t(end)); %Calculate MFB rate at each time step for plotting purposes
mfbi(i)=max(HRRhighres*dt/Qlhv,0)/NumberofHoles;
mfbstep = mfbstep+mfbi(i);
```

```
% First type of penetration calculation: threshold is M_threshold of maximum
% Momentum flux, which is approximately constant with x for steady jets, so it
% works well for penetration of the whole jet
ii = find(Mdot>M_threshold*max([Mdot0 Mdot]));
if isempty(ii)
   Penetration1(i) = 0;
   VLi=0;
else
   Penetration1(i) = x(ii(end));
   VLi=ii(end);
end

% Second type of penetration calculation:
% Calculate penetration using absolute equivalence ratio (e.g., for liquid length)
ii = find(phi.*gamma > phi_threshold);
if isempty(ii)
   Penetration2(i)=0;
   LLi=1;
else
   Penetration2(i)=x(ii(end));
   LLi=ii(end);
end
```

Plotting in Main Iteration

```
check=(mod(i,outputupdatestep) == 0)||(t(end)>2*dtCAstep*(imovie+1));
if(check)

   imovie=imovie+1
   %save a few select variables to dout
   if(imovie==1)
      dout.x{imovie} = x;
      dout.radius{imovie} = (x+x0)*tand(InjSpreadAngle/2);
      dout.alpha{imovie} = alpha;
   end

   dout.FA{imovie} = FA_Ratio;
   dout.Xf{imovie} = Xf;
   dout.t(imovie) = t(end)*1000;

   dout.S(imovie)= Penetration1(i);


   for hh=1:(radialsteps)
      for h=1:numel(x)
```

```
        phiradvar(hh,h)=phi(h)*gamma(h)*(1-((hh-1)/radialsteps)^alpha(h))^2;
        mftradvar(hh,h)=(1-((hh-1)/radialsteps)^alpha(h))^2;
        radial(hh,h)= (x(h)+x0)*tand(InjSpreadAngle/2)*(hh-1)/radialsteps;
    end
    axial(hh,:)=x;

end
S=sum(mftradvar);
for h=1:numel(x)

    mftradvar(:,h)=mft(h)*mftradvar(:,h)./S(h);

end

%removes previously burned mass (including transport) from total mass available
mfuradvar = mftradvar;

dout.phirad{imovie}=phiradvar;
dout.mftrad{imovie}=mftradvar;

for h = 1:numel(x)

    fuelburned = mfb(h);
    for hh= (radialsteps):-1:1
        fuelburnedpacket = min(mfuradvar(hh,h),fuelburned);
        mfuradvar(hh,h)=mftradvar(hh,h)- fuelburnedpacket;
        fuelburned = max(fuelburned - fuelburnedpacket,0);
    end
end




LPen=Penetration2(i)/total_length*numel(x);
VPen=Penetration1(i)/total_length*numel(x);
```

New Fuel distribution method based on Percent Heat Release

```
for h=1:numel(x)
    for hh=1:(radialsteps)
        percentHR(hh,h) = interp1(PhiInterp,percentInterp,phiradvar(hh,h)); %interpolate based on data on lines 103-104
    end
end

fuelleft=mfbstep; %sets the amount of fuel to be burned in timestep

iPen=cast(Penetration1(i)/total_length*numel(x),'int8');
```

```
mfbradvar=zeros(size(mfuradvar)); %initialize mfbradvar matrix
while fuelleft>0 %Is there fuel left to burn in timestep
   mfbtemp=mfuradvar.*percentHR;%multiply percent heat release by the amount of unburned fuel to determine avaliable fuel (mfbtemp)
   multiplier=(fuelleft/sum(sum(mfbtemp))); %multiplier used to scale fuel with the amount of fuel used in time step
   mfbtemp=mfbtemp*multiplier;
   if multiplier>1
      mfbtemp=min(mfuradvar,mfbtemp); %If multiplier is greater than 1, uses the minimum value to make sure fuel is not taken out where fuel is not avaliable
   end
   mfbradvar = mfbradvar + mfbtemp; %if multiple times thru while loop, this addes up
   mfuradvar = mfuradvar - mfbtemp; %if multiple times thru while loop, this keeps removing fuel
   fuelleft=max(mfbstep-sum(sum(mfbradvar)),0);%if fuel is left over, this keeps track and forces another calculation through while loop
end
%add newly burned fuel to mfb
mfb = mfb + sum(mfbradvar); %keeps track of total mass fuel burned for entire combustion event
```

Flame Temperature Interpolation

```
for h=1:numel(x)
   for hh=1:(radialsteps)
      FlameTradvar(hh,h) = interp2(FlameLookup.EGR,FlameLookup.Phi,FlameLookup.Temp,EGR,phiradvar(hh,h))+BulkTemp(SOIi+cast(t(end)*EngSpStep,'int16'))-700;
   end
end

FlameTradvar=FlameTradvar.*ceil(mfbradvar); %Makes temperature where no fuel is burned equal to zero
FlameTradvar(FlameTradvar==0)=nan;%BulkTemp(SOIi+cast(t(end)*EngSpStep,'int16'));
dout.FlameTrad{imovie}=FlameTradvar;
```

So far all the calculations have been for 1 half of the 1-D spray, these next lines copy and flip the matrix to create a complete 1-D spray

```
plotnumber=plotnumber+1;
axial2 = [axial(end:-1:2,:); axial];
radial2 = [-radial(end:-1:2,:); radial];
phiradvar2 = [phiradvar(end:-1:2,:) ; phiradvar];
mfbradvar2 = [mfbradvar(end:-1:2,:)/2;mfbradvar(1,:) ; mfbradvar(2:end,:)/2];
mfuradvar2 = [mfuradvar(end:-1:2,:)/2;mfuradvar(1,:) ; mfuradvar(2:end,:)/2];
mftradvar2 = [mftradvar(end:-1:2,:)/2;mftradvar(1,:) ; mftradvar(2:end,:)/2];
FlameTradvar2 = [FlameTradvar(end:-1:2,:) ; FlameTradvar];
```

Augments data for use in phi-T plots

```
mfb2_column=mfbradvar(:);%makes mfbradvar matrix into long column

Xedges =linspace(600,3500,bins); %creates Temperature steps for bins
```

```
Yedges = linspace(0,6,bins); %creates Phi steps for bins
X=discretize(FlameTradvar(:),Xedges); %Create Temperature bins using X edges
Y=discretize(phiradvar(:),Yedges); %Create Phi bins using Y edges
```

For Ignition Locations

```
for j=1:bins
   for i=1:bins
      mtotal_current(j,i)=mtotal_current(j,i)+ sum(mfb2_column(find(X==i & Y==j))); %Adds up mass that burns at Temp and Phi conditions
   end
end
radvar = phiradvar.*ceil(mfbradvar);
Contour_levels = Phi_Contour_levels;
zlim1 = [0 3];

if savefigure==1
   fprintf('%.3f/%.3f\r',t(end)*1000,Run_Time*1000)
end

if(sum(~isnan(FlameTradvar(:)))>0||(HRRhighres==0&&mfbhr(SOIi+cast(t(end)*EngSpStep,'int16'))>0))
```

```
   imovie1=imovie1+1;
   Tsprayradvar(:,:,imovie1)=FlameTradvar2;
   deltaT = (BulkTemp(SOIi+cast(t(end)*EngSpStep,'int16'))-BulkTemp(SOIi+cast(t(end)*EngSpStep,'int16')-1)); %Computes delta T for Bulk Gas Temperature
   for ii=imovie1-1:-1:1
      Tsprayradvar(:,:,ii)=Tsprayradvar(:,:,ii)+deltaT;
   end
```

For Post Mixing and Heating

```
   mfb_Hist(:,:,imovie1)=mfbradvar2;
   mfb2_column2 =mfb_Hist(:);
   X=discretize(Tsprayradvar(:),Xedges);
   Y=discretize(repmat(phiradvar2(:),imovie1,1),Yedges);

   for j=1:bins
      for i=1:bins
         mtotal_current2(j,i)=mtotal_current2(j,i)+ sum(mfb2_column2(find(X==i & Y==j)));
      end
   end


   FlameHist = [FlameHist FlameTradvar2(:)];
   PhiHist = [PhiHist phiradvar2(:)];
   FlameHist2 = [FlameHist2 Tsprayradvar(:)'];
```

```
        PhiHist2 = [PhiHist2 repmat(phiradvar2(:),size(Tsprayradvar,3),1)'];

        PhiHist2(isnan(FlameHist2))=[];
        FlameHist2(isnan(FlameHist2))=[];

        if(do_plot==1)
```

Full Screen 6 subplot

```
            figure(50)
            clf
%------------HRR and MFB plots with updating lines-------------------------
            subplot(3,2,2)
            hold on
            plot(CrAng(SOIi:EOCi),mfbhr(SOIi:EOCi),CrAng(SOIi:EOCi),HRR(SOIi:EOCi)/max(HRR))
            plot(CrAng(SOIi+cast(t(end)*EngSpStep,'int16')),HRRhighres/max(HRR),'r*')
            line([firstHRRpkdeg firstHRRpkdeg], [0 1])
            plot([EOIdeg EOIdeg], [0 1],'r')
            plot([CrAng(SOIi+cast(t(end)*EngSpStep,'int16')) CrAng(SOIi+cast(t(end)*EngSpStep,'int16'))], [0 1], 'm')
            plot([CrAng(SOIi) CrAng(EOCi)], [mfbhrhighres mfbhrhighres] ,'m')
            axis([SOIdeg EOCdeg 0 1])
            title('Normalized Heat Release and MFB','FontSize',(fontsize-2));
            xlabel('Crank Angle [deg aTDC]','FontSize',(fontsize-2));
            hold off
%-------------------Flame Temperature subplot-----------------------------
            subplot(3,2,1)
            handle=surf(axial2*1000,radial2*1000, Tsprayradvar(:,:,(imovie1)));
            handle=set(handle, 'edgecolor','none');
            %               xlim1 = [0 mod(x(VLi+1)*1000,50)-x(VLi+1)*1000 + 50];
            xlim1 = [0 max(ceil(VLi/25)*25,25)];
            ylim1 = [-20 20];

            zmax = max(max(FlameTradvar2));
            if ~isnan(zmax)&&~(zmax==0)&&~isinf(zmax)
               zlim1 = [0 max(max(FlameTradvar2))];
            else
               zlim1 = [0 1];
            end
            %axis equal;
            set(gca,'XLim',xlim1,'YLim',ylim1,'ZLim',zlim1);
            view(0,90)
            colorbar
            colormap(jet)
            caxis([ 600 3500])
            titlestr = sprintf('    Spray Temperature, MFT=%2.3f/%2.3f mg' ,sum(sum(mftradvar))*1e6,mfinj_tot(end)*1e6);

            xlabel('Axial Distance [mm]','FontSize',(fontsize-2));
```

```
        ylabel('Radial Distance [mm]','FontSize',(fontsize-2));
        title(titlestr,'FontSize',(fontsize-2));
        grid off
%-------------------------MFB subplot----------------------------------
        subplot(3,2,3)
        handle=surf(axial2*1000,radial2*1000, mfbradvar2);
        handle=set(handle, 'edgecolor','none');
        xlim1 = [0 max(ceil(VLi/25)*25,25)];
        ylim1 = [-20 20];
        set(gca,'XLim',xlim1,'YLim',ylim1,'ZLim',zlim1);
        view(0,90)
        colorbar
        titlestr = sprintf('mfbradvar,  MFB=%2.3f/%2.3f ug' ,sum(sum(mfbradvar))*1e9,mfbstep*1e9);

        xlabel('Axial Distance [mm]','FontSize',(fontsize-2));
        ylabel('Radial Distance [mm]','FontSize',(fontsize-2));
        title(titlestr,'FontSize',(fontsize-2));
        grid off
%-----------------------Ignition Location subplot----------------------
        subplot(3,2,4)
        contour3(Xedges,Yedges,mtotal_current,linspace(0,max(max(mtotal_current)),bins),'fill', 'on')
        colormap(jet)
        view(0,90)
        xlabel('Local Reaction Temperature (K)','FontSize',(fontsize-2));
        ylabel('  Equivalence Ratio (air based)','FontSize',(fontsize-2));
        set(gca,'Fontsize',(fontsize-2));
        grid off
%---------------------------MFU subplot----------------------------------
        subplot(3,2,5)
        handle= surf(axial2*1000,radial2*1000, mfuradvar2);
        handle=set(handle, 'edgecolor','none');
        %               xlim1 = [0 mod(x(VLi+1)*1000,50)-x(VLi+1)*1000 + 50];
        xlim1 = [0 max(ceil(VLi/25)*25,25)];
        ylim1 = [-20 20];
        set(gca,'XLim',xlim1,'YLim',ylim1,'ZLim',zlim1);
        view(0,90)
        colorbar

        titlestr = sprintf('mfuradvar, MFU=%2.3f/%2.3f mg' ,sum(sum(mfuradvar))*1e6,mfinj_tot(end)*1e6);

        xlabel('Axial Distance [mm]','FontSize',(fontsize-2));
        ylabel('Radial Distance [mm]','FontSize',(fontsize-2));
        title(titlestr,'FontSize',(fontsize-2));
        grid off
%------------------Post Mixing and Heating subplot-----------------------
        subplot(3,2,6)
        contour3(Xedges,Yedges,mtotal_current2,linspace(0,max(max(mtotal_current2)),bins),'fill', 'on')
```

```
colormap(jet)
view(0,90)
xlabel('Local Reaction Temperature (K)','FontSize',(fontsize-2));
ylabel('Equivalence Ratio (air based)  ','FontSize',(fontsize-2));
set(gca,'Fontsize',(fontsize-2))
grid off
```

Flame Temp. Standalone plot

```
figure(1000)
clf
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];
handle=surf(axial2*1000,radial2*1000, Tsprayradvar(:,:,(imovie1)));
handle=set(handle, 'edgecolor','none');
xlim1 = [0 max(ceil(VLi/25)*25,25)];
ylim1 = [-20 20];
set(gca,'XLim',xlim1,'YLim',ylim1,'ZLim',zlim1);
zmax = max(max(FlameTradvar2));
if ~isnan(zmax)&&~(zmax==0)&&~isinf(zmax)
   zlim1 = [0 max(max(FlameTradvar2))];
else
   zlim1 = [0 1];
end
view(0,90)
colorbar
caxis([ 600 3500])
xlabel('Axial Distance [mm]','FontSize',fontsize);
ylabel('Radial Distance [mm]','FontSize',fontsize);
%              title('Spray Temperature','FontSize',fontsize);
set(gca,'Fontsize',fontsize-2)
grid off
colormap(jet)
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   fname1 = sprintf('Tspray_%d.png',plotnumber);
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\'
date_str '\ T_Spray' '\' fname1])
end
```

Post Mixing and Heating Standalone plot

```
figure(1001)
clf
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];
contour3(Xedges,Yedges,mtotal_current2,linspace(0,max(max(mtotal_current2)),bins),'fill', 'on')
colormap(jet)
view(0,90)
xxxx=xlabel('Local Reaction Temperature (K)','FontSize',(fontsize));
P = get(xxxx,'Position');
set(xxxx,'Position',[P(1) P(2)-.1 P(3)])
ylabel('Equivalence Ratio (air based)  ','FontSize',(fontsize));
set(gca,'Fontsize',(fontsize-2))
grid off
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   fname2 = sprintf('Mixing_and_Heating_%d_mfb_%i.png',plotnumber,mfbhrhighres*100);
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\'
date_str '\ Mixing_and_Heating' '\' fname2])
end
```

MFB Standalone plot

```
mfbradvar2plot=mfbradvar2;
mfbradvar2plot(isnan(FlameTradvar2))=NaN; %used to make plot white when 0
figure(1002)
clf
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];
handle=surf(axial2*1000,radial2*1000, mfbradvar2plot);
handle=set(handle, 'edgecolor','none');
xlim1 = [0 max(ceil(VLi/25)*25,25)];
ylim1 = [-20 20];
set(gca,'XLim',xlim1,'YLim',ylim1,'ZLim',zlim1);
view(0,90)
colormap(jet)
colorbar
xlabel('Axial Distance [mm]','FontSize',(fontsize));
ylabel('Radial Distance [mm]','FontSize',(fontsize));
set(gca,'Fontsize',fontsize-2)
```

```
grid off
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   fname2 = sprintf('MFB_%d_mfb_%i.png',plotnumber,mfbhrhighres*100);
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\'
date_str '\ MFB' '\' fname2])
end
```

PHI Standalone plot

```
mfuradvar2plot=mfuradvar2;
mfuradvar2plot(mfuradvar2plot<1e-11)=NaN; %used to make plot white when 0
figure(1003)
clf
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 3.5 2.64];
handle=surf(axial2*1000,radial2*1000, ceil(mfuradvar2plot).*phiradvar2);
handle=set(handle, 'edgecolor','none');
xlim1 = [0 max(ceil(VLi/25)*25,25)];
ylim1 = [-20 20];
set(gca,'XLim',xlim1,'YLim',ylim1,'ZLim',zlim1);
view(0,90)
colormap(jet)
caxis([0 6])
colorbar
xlabel('Axial Distance [mm]','FontSize',(fontsize));
ylabel('Radial Distance [mm]','FontSize',(fontsize));
set(gca,'Fontsize',fontsize-2)
grid off
scalefactor=.9;
g = get(gca,'Position');
g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
g(3:4) = scalefactor*g(3:4);
set(gca,'Position',g);
if savefigure==1
   fname2 = sprintf('PHI_%d_mfb_%i.png',plotnumber,mfbhrhighres*100);
   print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\'
date_str '\ PHI' '\' fname2])
end
```

Ignition Location Standalone plot

```
            figure(1004)
            clf
            fig = gcf;
            fig.PaperUnits = 'inches';
            fig.PaperPosition = [0 0 3.5 2.64];
            contour3(Xedges,Yedges,mtotal_current,linspace(0,max(max(mtotal_current)),bins),'fill', 'on')
            colormap(jet)
            view(0,90)
            xxxx=xlabel('Local Reaction Temperature (K)','FontSize',(fontsize));
            P = get(xxxx,'Position');
            set(xxxx,'Position',[P(1) P(2)-.1 P(3)])
            ylabel('Equivalence Ratio (air based)  ','FontSize',(fontsize));
            set(gca,'Fontsize',(fontsize-2))
            grid off
            scalefactor=.9;
            g = get(gca,'Position');
            g(1:2) = g(1:2) + (1-scalefactor)/2*g(3:4);
            g(3:4) = scalefactor*g(3:4);
            set(gca,'Position',g);
            if savefigure==1
               fname2 = sprintf('Ignition_Locations_%d_mfb_%i.png',plotnumber,mfbhrhighres*100);
               print(gcf,'-dpng','-r600',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\'
date_str '\ Ignition_Locations' '\' fname2])
            end
            if savefigure==1
               fname = sprintf('%d.jpg',plotnumber);
               print(figure(50),'-djpeg',['Z:\Combustion Modeling Code\Figures and Comparisons\' filename '\' sprintf( 'Cond_%i',runselector) '\'
date_str '\ Full_Screen' '\' fname]);
            end

         end

      end
      mfbstep = 0;

   end

end



FlameHist2=FlameHist2(PhiHist2>0.1);
PhiHist2=PhiHist2(PhiHist2>0.1);
figure(96)
cloudPlot(FlameHist2(:), PhiHist2(:), [600 3500 0 6], [], [200 200]);
title(['Post Ignition Mixing/Heating - File =' filename ' Run #' sprintf('%d',runselector)])
figure(95)
```

```
cloudPlot(FlameHist, PhiHist, [600 3500 0 6], [], [200 200]);
title(['Ignition Locations - File =' filename ' Run #' sprintf('%d',runselector)])

%---------------------- End of Main Iteration --------------------------
toc
```

Published with MATLAB® R2015a