```
function varargout = Tool(varargin)
%Author: Tatiana D. Luna
%Igntition Intern Summer 2016
% TOOL MATLAB code for Tool.fig
%      TOOL, by itself, creates a new TOOL or raises the existing
%      singleton*.
%
%      H = TOOL returns the handle to a new TOOL or the handle to
%      the existing singleton*.
%
%      TOOL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TOOL.M with the given input arguments.
%
%      TOOL('Property','Value',...) creates a new TOOL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Tool_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Tool_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Tool

% Last Modified by GUIDE v2.5 22-Jul-2016 15:01:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Tool_OpeningFcn, ...
                   'gui_OutputFcn',  @Tool_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% --- Executes just before Tool is made visible.
function Tool_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Tool (see VARARGIN)
```

```
% Choose default command line output for Tool
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%setappdata will be used to share data among gui's
setappdata(0,'HandleMainGUI',hObject);
%setappdata(0, 'faultsData', hObject);

%placing the CUMMINS LOGO on the top left corner
[x,map]= imread('cummins_logo.jpg');
I2=imresize(x, [60 70]);
set(handles.togglebutton1,'cdata',I2);

% UIWAIT makes Tool wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%% --- Outputs from this function are returned to the command line.

function varargout = Tool_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% load(filename); %loads the file user has selected
% set(handles.fileuploadedtext,'string', {'File Uploaded'});

%% --- Executes on button press in togglebutton1 CUMMINS LOGO:
     %this is where the logo is, when pressed nothing will occur%
 function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of togglebutton1



%%
%%%%%%% Lists the FLEETS we have in listbox%%%%%%%%%%
% --- Executes during object creation, after setting all properties.
function fleetlistbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fleetlistbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
%this is what FLEETlist will display:
%TO ADD MORE FLEETS TO THE LIST INSERT BELOW, NOTE: will need to link
%folder to new fleet in fleetlistbox1_Callback function below and edit the
%rootdirectory path in seperate function Lookfor (line 25)

set(hObject,'String',{'ISCG';'ISL9G';'ISLGLNOx';'ISX12G';'Badger';'ISX12G_Old
er_Data'});


%% --- Executes on selection change in fleetlistbox1.
function fleetlistbox1_Callback(hObject, eventdata, handles)
% hObject    handle to fleetlistbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns fleetlistbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
fleetlistbox1
fleetitems = get(hObject,'String'); %stores the items in listbox
fleetindex_selected = get(hObject,'Value'); %gets the number value of item
selected
fleetitem_selected = fleetitems{fleetindex_selected}; %gets the string value
of item selected

hObject.UserData = fleetitem_selected; %storing the selected item in UserData

%THE if statements below is checking which fleet was selected then loads
%the folder with the function 'load_listbox(initial_dir, handles)'

%to INSESRT new fleet item selection, edit list above and edit this section
%example: if fleeindex_selected == 5 initialdir=('LOCATION of matdata of new
fleet')
if fleetindex_selected == 1
    initial_dir =
('\\cidconas15d\CIDC_EG500G_002$\ETD_Data\CWI\ISCG\MatData'); %location of
data for ISCG
    load_listbox(initial_dir,handles)

elseif fleetindex_selected == 2
    initial_dir =
('\\cidconas15d\CIDC_EG500G_002$\ETD_Data\CWI\ISL9G\MatData');
    load_listbox(initial_dir,handles)

elseif fleetindex_selected == 3
    initial_dir =
('\\cidconas15d\CIDC_EG500G_002$\ETD_Data\CWI\ISLGLNOx\MatData');
    load_listbox(initial_dir,handles)

elseif fleetindex_selected == 4
    initial_dir =
('\\cidconas15d\CIDC_EG500G_002$\ETD_Data\CWI\ISX12G\MatData');
    load_listbox(initial_dir,handles) %Calls function load_listbox below
elseif fleetindex_selected == 5
```

```
    initial_dir = ('\\cidconas15d\CIDC_EG500G_002$\ETD_Data\Badger\MatData');
    load_listbox(initial_dir, handles)
elseif fleetindex_selected == 6
    initial_dir = ('\\cidconas15d\CIDC_EG200G_002$\afp\ETD\Data');
    load_listbox(initial_dir, handles)
    %****** to insert new fleet make sure to ALSO edit directory path in
    %LookFor function( line 25)! ***********
end

%set( handles.listbox2,'String',list_units );

%%
%%WILL LOAD the folders with the units onto load box%%
function load_listbox(dir_path, handles)
cd (dir_path)
dir_struct = dir(dir_path);
Folders = dir_struct(~strncmpi('.', {dir_struct.name}, 1)); %gets rid of .
dots
dirfiles = [Folders.isdir];
subfiles = Folders(dirfiles);
[sorted_names,sorted_index] = sortrows({subfiles.name}'); %

handles.file_names = sorted_names;
handles.is_dir = [Folders.isdir];
handles.sorted_index = sorted_index;
guidata(handles.figure1,handles)
set(handles.listbox2,'String',handles.file_names,...
    'Value',1)

 %% --- Executes during object creation, after setting all properties.
 % listbox2 lists the folder names of the units
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



%% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox2 contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox2
index_selected = get(hObject,'Value');
list = get(hObject,'String');
```

```
item_selected = list{index_selected};

hObject.UserData = item_selected; %storing the selected item in UserData %try
and change to set()
%add button will call 'UserData'

%%
% --- Executes during object creation, after setting all properties.%
%Once File is uploaded text will change from ... to "File uploaded"%
function fileuploadedtext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fileuploadedtext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

%% STORES the selected ITEM from 'selected units to analyze' into 'UserData'
%
% --- Executes on selection change in ItemsSelectedlistbox.
function ItemsSelectedlistbox_Callback(hObject, eventdata, handles)
% hObject    handle to ItemsSelectedlistbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
ItemsSelectedlistbox contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
ItemsSelectedlistbox
index_selected = get(hObject,'Value');
list = get(hObject,'String');
item_selected = list{index_selected};
hObject.UserData = item_selected;
% openfileofselected = dir(item_selected); %seeing if can open file of
% selected , continue to try if not delete this
% display(openfileofselected);
%%



% --- Executes during object creation, after setting all properties.
function ItemsSelectedlistbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ItemsSelectedlistbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% ADD Button
% When pressed, gets the selected item from listbox 2, saved in 'UserData',
% and adds it to ItemsSelectedlistbox
% --- Executes on button press in addbutton.
function addbutton_Callback(hObject, eventdata, handles)
% hObject    handle to addbutton (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try
    %cellstr is converting the string data into cell matrix
    items = cellstr(get(handles.listbox2, 'UserData')); %gets the stored
selected item from unit list
catch
    unitslisted = get(handles.listbox2, 'String'); %for 2013 ed
    dex = get(handles.listbox2, 'Value');
    items = cellstr(unitslisted(dex));
end

try
alreadyinList = cellstr(get(handles.ItemsSelectedlistbox, 'String')); %gets
items from selected list
catch
    unitslisted = get(handles.ItemsSelectedlistbox, 'String'); %for 2013 ed
    dex = get(handles.ItemsSelectedlistbox, 'Value');
    alreadyinList = cellstr(unitslisted(dex));
end

Length = length(alreadyinList);

try
    fleetitem = cellstr(get(handles.fleetlistbox1,'UserData'));
catch
    unitslisted = get(handles.fleetlistbox1, 'String'); %for 2013 ed
    dex = get(handles.fleetlistbox1, 'Value');
    fleetitem = cellstr(unitslisted(dex));
end

fleetS = get(handles.addbutton, 'UserData');

%if the selected items list is empty, the first row will become the item
%chosen saved in 'UserData listbox2 handles'
if Length == 1 && isequal(alreadyinList(1,1),{''}) == 1
    alreadyinList = items;
    fleetS = fleetitem;
else
    alreadyinList(Length+1,1) = items;
    fleetS(Length+1, 1) = fleetitem;
end
set(handles.addbutton, 'UserData', fleetS);
% hObject.UserData = fleetS;
set(handles.ItemsSelectedlistbox, 'String', alreadyinList);
dataAlreadyinTable = get(handles.uitableDates, 'Data');

if isempty(dataAlreadyinTable) == 1

    dataAlreadyinTable = [alreadyinList str2double(datestr(date, 'yymmdd'))
str2double(datestr(date, 'yymmdd'))];
end

len = length(dataAlreadyinTable(:,2));
```

```
if isequal(dataAlreadyinTable(1,2), {''}) == 1 %if its empty inserts default
date
            Dates(1,1) = str2double(datestr(date, 'yymmdd')); %current date
is default date
            Dates(1,2) = str2double(datestr(date, 'yymmdd'));
end
for s= 1:length(dataAlreadyinTable(:,2)) %checks if any vaulues have been
edited
        if   isequal(dataAlreadyinTable(s,2), str2double(datestr(date,
'yymmdd'))) == 0 && isequal(dataAlreadyinTable(1,2), {''}) == 0
            Dates(s,1) = dataAlreadyinTable(s,2); %doesnt change edited data
        elseif len < Length
            Dates(s,1) = str2double(datestr(date, 'yymmdd'));
        end

        if isequal(dataAlreadyinTable(s,3), str2double(datestr(date,
'yymmdd'))) == 0 && isequal(dataAlreadyinTable(1,3), {''}) == 0
            Dates(s,2) = dataAlreadyinTable(s,3); %doesnt change edited data
        elseif len < Length
            Dates(s,2) = str2double(datestr(date, 'yymmdd'))
        end
end

if length(alreadyinList) > len
    Dates = cell2mat(Dates);
    Dates(len+1,1) = str2double(datestr(date, 'yymmdd'));
    Dates(len+1,2) = str2double(datestr(date, 'yymmdd'));
end

tableData = table2cell(table(alreadyinList, Dates(:,1),
Dates(:,2)));%couldn't morph into a matrix so made it into a table then
%converted it back to cell to store it in the uitable
set(handles.uitableDates, 'Data', tableData);
%%

% --- Executes on button press in removebutton.
function removebutton_Callback(hObject, eventdata, handles)
% hObject    handle to removebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MAAB
% handles    structure with handles and user data (see GUIDATA)

contents = cellstr(get(handles.ItemsSelectedlistbox, 'String'));
Fleets = get(handles.addbutton, 'UserData');

if length(contents)<1; return; end; %if already empty, do nothing

INDx = get(handles.ItemsSelectedlistbox,'Value');
contents(INDx)=[]; %remove the item
Fleets(INDx) = []; %remove the item- INDx should be same location for fleet
and units
set(handles.addbutton, 'UserData', Fleets);

Value=INDx-1;
if Value<1; Value=1;end %take care of exception
```

```
set(handles.ItemsSelectedlistbox,'String',contents,'Value',Value);
datainTable = get(handles.uitableDates, 'Data');
datainTable(INDx,:) = []; %removes the specified item in the table
set(handles.uitableDates, 'Data', datainTable); %sets new data onto table


%%

% --- Executes during object creation, after setting all properties.
function uitableDates_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uitableDates (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
%%

% --- Executes when entered data in editable cell(s) in uitableDates.
function uitableDates_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitableDates (see GCBO)
% eventdata  structure with the following fields (see
MATLAB.UI.CONTROL.TABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty
if Data was not changed
%   Error: error string when failed to convert EditData to appropriate value
for Data
% handles    structure with handles and user data (see GUIDATA)

tableINFO=get(handles.uitableDates, 'Data');
[startdates, stringStartdates] = checkdates(tableINFO(:,2)); %runs function
and checks to see if it is a date entry
[enddates, stringEnddates] = checkdates(tableINFO(:,3)); %stops after here
DATES = [tableINFO(:,1) startdates enddates];
set(handles.uitableDates, 'Data' , DATES);
% datesInString = {stringStartdates, stringEnddates}; %will delete
% set(handles.Trackpushbutton, 'UserData', datesInString);

function uitableDates_CellSelectionCallback(hObject, eventdata, handles)
% hObject    handle to uitableDates (see GCBO)
% eventdata  structure with the following fields (see
MATLAB.UI.CONTROL.TABLE)


% --- Executes on button press in Trackpushbutton.
function Trackpushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Trackpushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tic
set(handles.Trackpushbutton, 'BackgroundColor', [1, .4, .4]);
set(handles.Trackpushbutton,'string', {'Loading'});
set(handles.figure1, 'pointer', 'watch') %sets pointer to circular loading
drawnow;
pause(2) %gives it time to change color
```

```
try

    Input = get(handles.uitableDates, 'Data'); %gets data from table which
has units, startdates, enddates
    fleets = get(handles.addbutton, 'UserData'); %engines
    %DateStart= {}; DateEnd={};
    [~, DateStart] = checkdates(Input(:,2));
    [~, DateEnd]= checkdates(Input(:,3));
    DateStart = cell(DateStart); DateEnd = cell(DateEnd);

    targethours = str2num(get(handles.editHours, 'string'));
    targetmiles = str2num(get(handles.editMiles, 'string'));
    targetvalues = [targethours, targetmiles];
    %initializing variables:
    totals =zeros(length(Input(:,1)),2); targethrs =
zeros(length(Input(:,1)),1);
    targetMiles = zeros(length(Input(:,1)),1);
    estMiles=cell(length(Input(:,1)),1); estHours =
cell(length(Input(:,1)),1);

    for it = 1:length(Input(:,1))
        targethrs(it,1) = targethours; %resizing in order to put on table
        targetMiles(it,1) = targetmiles;
        [fileLocations, lastday] = LookFor(fleets(it,1), Input(it,1),
Input(it,2), Input(it,3));
        [totals(it,:), estMiles{it,1}, estHours{it,1}]= track(fileLocations,
lastday, targetvalues);
        clear fileLocations lastday
    end

    %set(handles.Trackpushbutton, 'UserData', fileLocations); %idk y i put
    %this here delete?

    if length(Input(:,1)) == 1
        T= table(fleets, Input(:,1), DateStart(:,1), DateEnd(:,1),
totals(:,1), totals(:,2),...
        targetmiles, estMiles, targethours, estHours);
    else
        T= table(fleets, Input(:,1), DateStart{:,1}, DateEnd{:,1},
totals(:,1), totals(:,2),...
        targetMiles, estMiles, targethrs, estHours);
    end

    d = table2cell(T);

    setappdata(0   , 'Tool'     , gcf);
    setappdata(gcf,    'd', d);

    cname1 = 'Estimate Date to reach Target Miles';
    cname2 = 'Estimate Date to reach Target Hours';

    cnames = {'Engine', 'Unit','Start Date','End Date', ...
        'Total Miles', 'Total Hours','Target Miles ', cname1, 'Target Hours'
cname2};
    name = ('Estimating Dates Table 1');
```

```
    % tablegui; %opens the table gui
    set(tablegui,'Name', name,'NumberTitle','off') %set figure name
    uitable('Data', d, 'ColumnName',cnames, 'Position', [16 61 861 353]);
    HandleMainGUI=getappdata(0,'HandleMainGUI');
    setappdata(HandleMainGUI,'ESTdatesData', d); %tablegui will be able to
collect this data
    set(handles.Trackpushbutton, 'BackgroundColor', [0, .8, .4]);
    set(handles.Trackpushbutton,'string', {'Track'});
    pause(1)
    set(handles.figure1, 'pointer', 'arrow')
catch
    warning('Error Occured');
    set(handles.figure1, 'pointer', 'arrow');%returns pointer to arrow
    set(handles.Trackpushbutton,'string', {'ERROR'});
end
enddd = toc
%%
function editHours_Callback(~, eventdata, handles)
% hObject    handle to editHours (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editHours as text
%        str2double(get(hObject,'String')) returns contents of editHours as a
double


% --- Executes during object creation, after setting all properties.
function editHours_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editHours (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editMiles_Callback(hObject, eventdata, handles)
% hObject    handle to editMiles (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editMiles as text
%        str2double(get(hObject,'String')) returns contents of editMiles as a
double


% --- Executes during object creation, after setting all properties.
function editMiles_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editMiles (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
Trackpushbutton.
function Trackpushbutton_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to Trackpushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on key press with focus on Trackpushbutton and none of its
controls.
function Trackpushbutton_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to Trackpushbutton (see GCBO)
% eventdata  structure with the following fields (see
MATLAB.UI.CONTROL.UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)

%%
% --- Executes on button press in FaultCodespushbutton.
function FaultCodespushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to FaultCodespushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

tic
set(handles.figure1, 'pointer', 'watch');
drawnow;
set(handles.FaultCodespushbutton, 'BackgroundColor', [1, .4, .4]);
set(handles.FaultCodespushbutton,'string', {'Loading'});
pause(1)
try
    Input = get(handles.uitableDates, 'Data'); %gets data from table which
has units, startdates, enddates
    fleets = get(handles.addbutton, 'UserData'); %engines

    for it = 1:length(Input(:,1))
        [fileLocations, ~] = LookFor(fleets(it,1), Input(it,1), Input(it,2),
Input(it,3));
        check_InActive(fileLocations, Input(it,2), Input(it,3), it);
    end
```

```
    set(handles.figure1, 'pointer', 'arrow')
    set(handles.FaultCodespushbutton, 'BackgroundColor', [0, .8, .4]);
    set(handles.FaultCodespushbutton,'string', {'Examine'});
catch
    warning('Error Occured');
    set(handles.figure1, 'pointer', 'arrow')
    set(handles.FaultCodespushbutton,'string', {'Error'});
end
toc
%%
% --- Executes on selection change in FieldTestlistbox.
function FieldTestlistbox_Callback(hObject, eventdata, handles)
% hObject    handle to FieldTestlistbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns FieldTestlistbox
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
FieldTestlistbox
index_selected = get(hObject,'Value');
list = get(hObject,'String');
item_selected = list{index_selected};
getfieldtests = get(handles.FieldTestlistbox, 'UserData'); %excel sheet data
rowsofselectedtest = find(strcmp(getfieldtests(:,2), item_selected));

if index_selected == 1
    fprintf('\n Please select a field test. \n');
    return;
end

for x = 1:length(rowsofselectedtest)
    row = rowsofselectedtest(x,1);
  if strcmp(getfieldtests(row,20),'') == 0
    if x == 1
        unitswithfolders(1,1) = getfieldtests(row,20);
        installeddates(1,1) = getfieldtests(row,10);
        removeddates(1,1) = getfieldtests(row,11);
        fleets(1,1) = getfieldtests(row,1);
    else
        %L=length(unitswithfolders);
        if exist('unitswithfolders', 'var')==1
            unitswithfolders(end+1,1) = getfieldtests(row,20);
            installeddates(end+1,1) = getfieldtests(row,10);
            removeddates(end+1,1) = getfieldtests(row,11);
            fleets(end+1,1) = getfieldtests(row,1);
        else
            unitswithfolders(1,1) = getfieldtests(row,20);
            installeddates(1,1) = getfieldtests(row,10);
            removeddates(1,1) = getfieldtests(row,11);
            fleets(1,1) = getfieldtests(row,1);
        end
    end
  end
end
```

```
if exist('unitswithfolders','var') == 0
      fprintf('\n');
      warning('No avaliable folders for units, please check column T in
\\cidconas15d\CIDC_EG200G_002$\afp\Spark Plug Analysis\Field Test
Logger.xlsx');
      return;
end

installeddates = num2cell(str2num(datestr(installeddates, 'yymmdd')));

for Q = 1: length(removeddates)
    if isequal(removeddates{Q,1},'') == 1
        removedDates(Q,1) = str2double(datestr(date, 'yymmdd'));
    else
        removedDates(Q,1) = str2num(datestr(removeddates{Q,1}, 'yymmdd'));
    end
end
removedDates = num2cell(removedDates);
Fleets = get(handles.addbutton, 'UserData');
if isempty(Fleets) == 1
    Fleets = fleets;
else
    Fleets = [Fleets; fleets];
end
set(handles.addbutton, 'UserData',Fleets); %%% will have to change for

alreadyinList = cellstr(get(handles.ItemsSelectedlistbox, 'String')); %gets
items from selected list
Length = length(alreadyinList);
for y = 1:length(unitswithfolders)
    %if the selected items list is empty, the first row will become the item
    %chosen saved in 'UserData listbox2 handles'
    if Length == 1 && isequal(alreadyinList(1,1),{''}) == 1
        alreadyinList(y,1) = unitswithfolders(y,1);

    else
        alreadyinList(end+1,1) = unitswithfolders(y,1);
    end

end

set(handles.ItemsSelectedlistbox, 'String', alreadyinList);
dataAlreadyinTable = get(handles.uitableDates, 'Data');

if isempty(dataAlreadyinTable) == 1
    dataAlreadyinTable = [alreadyinList installeddates removedDates];
elseif isequal(dataAlreadyinTable(1,2), {''}) == 1 %if its empty
    dataAlreadyinTable = [alreadyinList installeddates removedDates];
else
    morphing1 =[dataAlreadyinTable(:,2); installeddates];
    morphing2 = [dataAlreadyinTable(:,3); removedDates];
    dataAlreadyinTable = [alreadyinList, morphing1, morphing2];
end
```

```
tableData =
dataAlreadyinTable;%table2cell(table(dataAlreadyinTable));%couldn't morph
into a matrix so made it into a table then
%converted it back to cell to store it in the uitable
set(handles.uitableDates, 'Data', tableData);

%%
% --- Executes during object creation, after setting all properties.
function FieldTestlistbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FieldTestlistbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
filepathtoLogger = ['\\cidconas15d\CIDC_EG200G_002$\afp\Spark Plug
Analysis\Field Test Logger.xlsx'];
[num,txt,raw] = xlsread(filepathtoLogger, 'Sheet1');
fieldTests = unique(txt(:,2)); %deleting all the repeats, gets the fields
tests
rowwithlabel = find(strcmp(fieldTests, 'Field Test'),1);
fieldTests(rowwithlabel,:) = [];
listthis = ['Select a Field Test:'; fieldTests];
set(hObject,'String', listthis);
hObject.UserData = txt;

%%
% --- Executes on button press in SVDcheckbox.
function SVDcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to SVDcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of SVDcheckbox


% --- Executes on button press in dailySVDcheckbox.
function dailySVDcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to dailySVDcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of dailySVDcheckbox


% --- Executes on button press in scatterplotSVDcheckbox.
function scatterplotSVDcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to scatterplotSVDcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of scatterplotSVDcheckbox


% --- Executes on button press in PLOTpushbutton.
function PLOTpushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to PLOTpushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


savegif = get(handles.savegifcheckbox, 'Value');
if savegif == 1
    gpath = uigetdir;
end

save = get(handles.save2pptcheckbox, 'Value');
if save == 1
    [file, path] = uigetfile('*.ppt*');
end

tic
set(handles.PLOTpushbutton, 'BackgroundColor', [1, .4, .4]);
set(handles.PLOTpushbutton,'string', {'Loading'});
set(handles.figure1, 'pointer', 'watch')
drawnow;

Input = get(handles.uitableDates, 'Data'); %gets data from table which has
units, startdates, enddates
fleets = get(handles.addbutton, 'UserData'); %engines

%******Obtaining the checked values; which plots were selected; 0 = no; 1 =
plot***
%plots that use average voltage
scatterDaily = get(handles.scatterTorqSpeedcheckbox, 'Value'); %this is just
speed vs torque
scatterSVDdaily = get(handles.scatterplotSVDcheckbox,'Value'); %voltage
ranges seperated
scatterSVD = get(handles.scatterSVDcheckbox , 'Value'); %voltage ranges
combined
svd = get(handles.SVDcheckbox,'Value'); %plot svd vs time for entire time
interval w/ torque&speed
Dailysvd = get(handles.dailySVDcheckbox,'Value'); %svd vs time daily
extraSVD = get(handles.extrapolateSVDcheckbox, 'Value'); %extrapolate svd
avgSVDnoTor = get(handles.avgSVDcylinderscheckbox, 'Value');%Svd for each
cylinder no torque&speed plot
maxEnv = get(handles.maxEnvelopecheckbox, 'Value'); %max envelope of average
SVD

%plots that use 200ms voltage
extraNorm = get(handles.extraNormcheckbox, 'Value'); %extrapolate normalized
voltage
normalV = get(handles.scatternormalizedcheckbox, 'Value'); %scatter plot of
normalized V
normalVRange = get(handles.normalizedRangescheckbox, 'Value'); %scatter plot
voltage ranges seperated
```

```
dailynormal = get(handles.normalvsDailytimecheckbox, 'Value'); %daily norm V
vs time
instantSVD = get(handles.instantSVDcheckbox, 'Value'); %200ms Voltage vs time
normalsvdtotaltime = get(handles.normalSVDtotalcheckbox,'Value'); %total time
interval vs normal V

Dailymisfire = get(handles.dailyMisfirecheckbox, 'Value');
misfire = get(handles.Misfirecheckbox, 'Value');

%Duty cycle plots
dutyDaily = get(handles.DutyPlotcheckbox, 'Value'); %daily plots of speed vs
torque
duty = get(handles.dutyCombinedcheckbox, 'Value');
dailymesh = get(handles.dailymeshcheckbox, 'Value');
massduty = get(handles.MassAirFcheckbox, 'Value');
MAFdailyduty = get(handles.DailyMAFcheckbox, 'Value');
MAFmeshdaily = get(handles.DailyMeshMAFcheckbox, 'Value');

%svd vs distance
distSVD = get(handles.SVD_milescheckbox, 'Value');
extDistSvd = get(handles.extractmilescheckbox, 'Value');
extractMilesNorm = get(handles.extractNORMmilescheckbox, 'Value');

wlamp = get(handles.warninglampcheckbox, 'Value');

for it = 1:length(Input(:,1))
    [fileLocations, ~] = LookFor(fleets(it,1), Input(it,1), Input(it,2),
Input(it,3));
    engine = fleets{it,1};
    %these variables are for plotting total time intervals
    TotalInterval_SVD_Cyl_1_Voltage = []; TotalInterval_SVD_Cyl_2_Voltage =
[];
    TotalInterval_SVD_Cyl_3_Voltage = []; TotalInterval_SVD_Cyl_4_Voltage =
[];
    TotalInterval_SVD_Cyl_5_Voltage = []; TotalInterval_SVD_Cyl_6_Voltage =
[];
    TotalTorque = []; Totalspeed = [];
    total_run_time = 0; totalmeters = [];
    %variables for calculating misfires for total time interval
    total_Misfire_1 = []; total_Misfire_2 = []; total_Misfire_5 = [];
    total_Misfire_3 = []; total_Misfire_4 = []; total_Misfire_6 = [];
    total_MisfireActive = []; total_IMDALL = [];MTotalTorque=[]; MTotalspeed
=[];
    Mrun_time = [];
    %variables for ISL misfire
    totalMisfire_1=[];totalMisfire_2=[]; totalMisfire_3=[];
    totalMisfire_4=[]; totalMisfire_5=[];totalMisfire_6=[];
    %Variables for normalizations
    TotalSTC_Cyl_1_Advance=[]; TotalIntake_Manifold_Pressure=[];
    TotalSTC_Cyl_1_Knock = []; TotalSTC_Cyl_2_Knock=[];
    TotalSTC_Cyl_3_Knock=[];TotalSTC_Cyl_4_Knock=[];
    TotalSTC_Cyl_5_Knock=[];TotalSTC_Cyl_6_Knock=[]; NTotalmotoring=[];
    NTotalInterval_SVD_Cyl_1_Voltage = []; NTotalInterval_SVD_Cyl_2_Voltage =
[];
    NTotalInterval_SVD_Cyl_3_Voltage = []; NTotalInterval_SVD_Cyl_4_Voltage =
[];
```

```
    NTotalInterval_SVD_Cyl_5_Voltage = []; NTotalInterval_SVD_Cyl_6_Voltage =
[];
    NTotalTorque = []; NTotalspeed = []; Ntotal_run_time = 0; NTotalmeters
=[];
    %Variable for duty plots
    dTotalTorque = []; dTotalSpeed =[]; mTotalMAF =[]; mTotalSpeed =[];

    for x = 1:length(fileLocations(:,1))

    try
%*********Loading instantanous voltage 'Voltage_200ms' only for
normalizaiton%
        if (extraNorm==1 || normalV ==1 || normalVRange==1 || dailynormal ==
1 ||...
                instantSVD ==1 || normalsvdtotaltime ==1 ||extractMilesNorm
==1) && (svd==0 && ...
                Dailysvd == 0 && extraSVD == 0 && scatterDaily ==0 &&
scatterSVD ==0 ...
                && scatterSVDdaily ==0 && avgSVDnoTor == 0 && maxEnv==0 &&
distSVD ==0 &&...
                extDistSvd == 0 && wlamp ==0)

            if strcmpi(engine(1:3),'ISX') == 1

                if Dailymisfire == 0 && misfire == 0
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time_200ms',...
                   'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms' ,...
                    'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...
                   'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms',...
                   'ENGN_Final_Torque_Cmd_200ms',
'Engine_Speed_200ms','STC_Cyl_1_Advance_200ms',...

'Intake_Manifold_Pressure_200ms','STC_Cyl_1_KnockOffset_200ms',...

'STC_Cyl_2_KnockOffset_200ms','STC_Cyl_3_KnockOffset_200ms',...

'STC_Cyl_4_KnockOffset_200ms','STC_Cyl_5_KnockOffset_200ms',...

'STC_Cyl_6_KnockOffset_200ms','MTR_Active_200ms','Mass_Air_Flow*',...
                   'TI_Vehicle_Total_Engine_Dist')

                   if exist( 'ENGN_Final_Torque_Cmd_200ms','var') == 0 &&
...
                       exist( 'Engine_Speed_200ms','var') == 0
                    load([fileLocations{x,2} '\' fileLocations{x,1}],...
                        'ENGN_Final_Torque_Cmd_Screen_1',
'Engine_Speed_Screen_1')
                   ENGN_Final_Torque_Cmd_200ms =
ENGN_Final_Torque_Cmd_Screen_1;
                   Engine_Speed_200ms = Engine_Speed_Screen_1;
                   clear Engine_Speed_Screen_1
ENGN_Final_Torque_Cmd_Screen_1
                   elseif exist( 'ENGN_Final_Torque_Cmd_200ms','var') == 0
                   load([fileLocations{x,2} '\' fileLocations{x,1}],...
                        'ENGN_Final_Torque_Cmd_Screen_1')
```

```
                    ENGN_Final_Torque_Cmd_200ms =
ENGN_Final_Torque_Cmd_Screen_1;
                elseif  exist( 'Engine_Speed_200ms','var') == 0
                    load([fileLocations{x,2} '\' fileLocations{x,1}],...
                        'Engine_Speed_Screen_1')
                Engine_Speed_200ms = Engine_Speed_Screen_1;
                end
            if exist('STC_Cyl_1_KnockOffset_200ms', 'var') == 0
            %if doesnt exist load avg svd
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'STC_Cyl_1_Advance',...
                'STC_Cyl_*', 'Intake_Manifold_Pressure')
%                   STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance;
%                   STC_Cyl_1_KnockOffset_200ms = STC_Cyl_1_KnockOffset;
%                   STC_Cyl_2_KnockOffset_200ms = STC_Cyl_2_KnockOffset;
%                   STC_Cyl_3_KnockOffset_200ms = STC_Cyl_3_KnockOffset;
%                   STC_Cyl_4_KnockOffset_200ms = STC_Cyl_4_KnockOffset;
%                   STC_Cyl_5_KnockOffset_200ms = STC_Cyl_5_KnockOffset;
%                   STC_Cyl_6_KnockOffset_200ms = STC_Cyl_6_KnockOffset;

            end

            else
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time_200ms',...
                'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms'
,'MTR_Active_200ms',...
                 'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...
                'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms',...
                'ENGN_Final_Torque_Cmd_200ms', 'Engine_Speed_200ms',...
                'IMD_Cyl_1_MisfirePercent','IMD_Cyl_2_MisfirePercent'
,...
                'IMD_Cyl_3_MisfirePercent','IMD_Cyl_4_MisfirePercent',...
                'IMD_Cyl_5_MisfirePercent','IMD_Cyl_6_MisfirePercent',...

'IMD_Active','IMD_Cyl_All_MisfirePercent','STC_Cyl_1_Advance_200ms',...

'Intake_Manifold_Pressure_200ms','STC_Cyl_1_KnockOffset_200ms',...

'STC_Cyl_2_KnockOffset_200ms','STC_Cyl_3_KnockOffset_200ms',...

'STC_Cyl_4_KnockOffset_200ms','STC_Cyl_5_KnockOffset_200ms',...
                'STC_Cyl_6_KnockOffset_200ms', 'ECM_Run_Time',
'Net_Engine_Torque',...

'Engine_Speed','Mass_Air_Flow*','TI_Vehicle_Total_Engine_Dist')
            end



        elseif strcmpi(engine(1:3),'ISL') == 1
            if Dailymisfire == 0 && misfire == 0
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time_Screen_1',...
                'SVD_Cyl_1_Voltage_Screen_1','SVD_Cyl_2_Voltage_Screen_1'
,...
```

```
'SVD_Cyl_3_Voltage_Screen_1','SVD_Cyl_4_Voltage_Screen_1',...

'SVD_Cyl_5_Voltage_Screen_1','SVD_Cyl_6_Voltage_Screen_1',...
                    'ENGN_Final_Torque_Cmd_Screen_1',
'Engine_Speed_Screen_1',...

'STC_Cyl_1_Advance_Screen_1','Intake_Manifold_Pressure_Screen_1',...

'STC_Cyl_1_KnockOffset_Screen_1','STC_Cyl_2_KnockOffset_Screen_1'...

,'STC_Cyl_3_KnockOffset_Screen_1','STC_Cyl_4_KnockOffset_Screen_1',...

'STC_Cyl_5_KnockOffset_Screen_1','STC_Cyl_6_KnockOffset_Screen_1',...

'MTR_Active_Screen_1','Mass_Air_Flow*','TI_Vehicle_Total_Engine_Dist')
                else
                        load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time_Screen_1',...
                  'SVD_Cyl_1_Voltage_Screen_1','SVD_Cyl_2_Voltage_Screen_1'
,...

'SVD_Cyl_3_Voltage_Screen_1','SVD_Cyl_4_Voltage_Screen_1',...

'SVD_Cyl_5_Voltage_Screen_1','SVD_Cyl_6_Voltage_Screen_1',...
                    'ENGN_Final_Torque_Cmd_Screen_1',
'Engine_Speed_Screen_1',...

'STC_Cyl_1_Advance_Screen_1','Intake_Manifold_Pressure_Screen_1',...

'STC_Cyl_1_KnockOffset_Screen_1','STC_Cyl_2_KnockOffset_Screen_1'...

,'STC_Cyl_3_KnockOffset_Screen_1','STC_Cyl_4_KnockOffset_Screen_1',...

'STC_Cyl_5_KnockOffset_Screen_1','STC_Cyl_6_KnockOffset_Screen_1',...
                    'SBMD_IntMisfirePercentCyl1',
'SBMD_IntMisfirePercentCyl2', ...
                    'SBMD_IntMisfirePercentCyl3',
'SBMD_IntMisfirePercentCyl4', ...
                    'SBMD_IntMisfirePercentCyl5',
'SBMD_IntMisfirePercentCyl6',...
                    'SBMD_ContMisfirePercentCyl1',
'SBMD_ContMisfirePercentCyl2', ...
                    'SBMD_ContMisfirePercentCyl3',
'SBMD_ContMisfirePercentCyl4',...

'SBMD_ContMisfirePercentCyl5','SBMD_ContMisfirePercentCyl6',...

'MTR_Active_Screen_1','Mass_Air_Flow*','TI_Vehicle_Total_Engine_Dist')
                end

                if exist( 'ENGN_Final_Torque_Cmd_Screen_1','var') == 0 && ...
                            exist( 'Engine_Speed_Screen_1','var') == 0 &&
exist('STC_Cyl_1_KnockOffset_Screen_1', 'var') == 0
                     load([fileLocations{x,2} '\' fileLocations{x,1}],...
                         'ENGN_Final_Torque_Cmd_200ms',
'Engine_Speed_200ms',...
```

```
'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms',...
                    'MTR_Active_200ms','Mass_Air_Flow*')

                elseif exist( 'ENGN_Final_Torque_Cmd_Screen_1','var') == 0 ||
...
                        exist( 'Engine_Speed_Screen_1','var') == 0
                    load([fileLocations{x,2} '\' fileLocations{x,1}],...
                         'ENGN_Final_Torque_Cmd_200ms', 'Engine_Speed_200ms')
                elseif exist('STC_Cyl_1_KnockOffset_Screen_1', 'var') == 0
                    load([fileLocations{x,2} '\' fileLocations{x,1}],...

'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms')
                else

                    ECM_Run_Time_200ms = ECM_Run_Time_Screen_1;
                    SVD_Cyl_1_Voltage_200ms = SVD_Cyl_1_Voltage_Screen_1;
                    SVD_Cyl_2_Voltage_200ms = SVD_Cyl_2_Voltage_Screen_1;
                    SVD_Cyl_3_Voltage_200ms = SVD_Cyl_3_Voltage_Screen_1;
                    SVD_Cyl_4_Voltage_200ms = SVD_Cyl_4_Voltage_Screen_1;
                    SVD_Cyl_5_Voltage_200ms = SVD_Cyl_5_Voltage_Screen_1;
                    SVD_Cyl_6_Voltage_200ms = SVD_Cyl_6_Voltage_Screen_1;
                    ENGN_Final_Torque_Cmd_200ms =
ENGN_Final_Torque_Cmd_Screen_1;
                    Engine_Speed_200ms = Engine_Speed_Screen_1;
                    STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance_Screen_1;
                    Intake_Manifold_Pressure_200ms =
Intake_Manifold_Pressure_Screen_1;
                    STC_Cyl_1_KnockOffset_200ms =
STC_Cyl_1_KnockOffset_Screen_1;
                    STC_Cyl_2_KnockOffset_200ms =
STC_Cyl_2_KnockOffset_Screen_1;
                    STC_Cyl_3_KnockOffset_200ms =
STC_Cyl_3_KnockOffset_Screen_1;
                    STC_Cyl_4_KnockOffset_200ms =
STC_Cyl_4_KnockOffset_Screen_1;
                    STC_Cyl_5_KnockOffset_200ms =
STC_Cyl_5_KnockOffset_Screen_1;
                    STC_Cyl_6_KnockOffset_200ms =
STC_Cyl_6_KnockOffset_Screen_1;
                    MTR_Active_200ms = MTR_Active_Screen_1;
                    Mass_Air_Flow_200ms = Mass_Air_Flow_Screen_1;
                end
```

```
            elseif strcmpi(engine,'badger') == 1
               if Dailymisfire == 0 && misfire == 0
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time_200ms',...
                    'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms' ,...
                    'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...

'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms','TI_Vehicle_Total_Engine_
Dist',...
                    'CBM_NetTorqueDemand_200ms',
'Engine_Speed_200ms','MTR_Active_200ms',...

'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms','Mass_Air_Flow*')
                    if exist('CBM_NetTorqueDemand_200ms','var') == 1
                        ENGN_Final_Torque_Cmd_200ms =
CBM_NetTorqueDemand_200ms;
                    end

                else
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time_200ms',...
                    'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms' ,...
                    'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...
                    'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms',...
                    'CBM_NetTorqueDemand_200ms',
'Engine_Speed_200ms','TI_Vehicle_Total_Engine_Dist',...

'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms',...

'OBD_Misfire_Cyl1_CAT_Ratio_200ms','OBD_Misfire_Cyl2_CAT_Ratio_200ms',...

'OBD_Misfire_Cyl3_CAT_Ratio_200ms','OBD_Misfire_Cyl4_CAT_Ratio_200ms',...

'OBD_Misfire_Cyl5_CAT_Ratio_200ms','OBD_Misfire_Cyl6_CAT_Ratio_200ms',...

'Net_Engine_Torque','Engine_Speed','MTR_Active_200ms','Mass_Air_Flow*')
                    if exist('CBM_NetTorqueDemand_200ms','var') == 1
                        ENGN_Final_Torque_Cmd_200ms =
CBM_NetTorqueDemand_200ms;
                    end

                end

                if exist( 'SVD_Cyl_1_Voltage_200ms','var') == 0
```

```
                    load([fileLocations{x,2} '\' fileLocations{x,1}], ...
'SVD_Cyl_1_Voltage_Screen_1','SVD_Cyl_2_Voltage_Screen_1',...
                        'SVD_Cyl_3_Voltage_Screen_1',
'SVD_Cyl_4_Voltage_Screen_1',...
                        'SVD_Cyl_5_Voltage_Screen_1',
'SVD_Cyl_6_Voltage_Screen_1')
                    if exist('SVD_Cyl_1_Voltage_Screen_1', 'var') == 0
                        load([fileLocations{x,2} '\' fileLocations{x,1}], ...
                            'SVD_Cyl_1_Voltage','SVD_Cyl_2_Voltage',...
                            'SVD_Cyl_3_Voltage', 'SVD_Cyl_4_Voltage',...
                            'SVD_Cyl_5_Voltage', 'SVD_Cyl_6_Voltage')
                        SVD_Cyl_1_Voltage_200ms = SVD_Cyl_1_Voltage;
                        SVD_Cyl_2_Voltage_200ms = SVD_Cyl_2_Voltage;
                        SVD_Cyl_3_Voltage_200ms = SVD_Cyl_3_Voltage;
                        SVD_Cyl_4_Voltage_200ms = SVD_Cyl_4_Voltage;
                        SVD_Cyl_5_Voltage_200ms = SVD_Cyl_5_Voltage;
                        SVD_Cyl_6_Voltage_200ms = SVD_Cyl_6_Voltage;
                    else

                        SVD_Cyl_1_Voltage_200ms = SVD_Cyl_1_Voltage_Screen_1;
                        SVD_Cyl_2_Voltage_200ms = SVD_Cyl_2_Voltage_Screen_1;
                        SVD_Cyl_3_Voltage_200ms = SVD_Cyl_3_Voltage_Screen_1;
                        SVD_Cyl_4_Voltage_200ms = SVD_Cyl_4_Voltage_Screen_1;
                        SVD_Cyl_5_Voltage_200ms = SVD_Cyl_5_Voltage_Screen_1;
                        SVD_Cyl_6_Voltage_200ms = SVD_Cyl_6_Voltage_Screen_1;
                    end
                end
            end

   %*****Loading average voltage only; normalized svd was not selected******%
        elseif (extraNorm==0 && normalV ==0 && normalVRange==0 && dailynormal
== 0 && instantSVD==0 &&normalsvdtotaltime==0) ...
                && ( svd==1 || Dailysvd == 1 || extraSVD == 1 || scatterDaily
==1 || scatterSVD == 1 ...
                || scatterSVDdaily ==1 || avgSVDnoTor == 1 || maxEnv==1 ||
distSVD ==1 || extDistSvd==1)
            if Dailymisfire == 0 && misfire == 0
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...
                    'Net_Engine_Torque',
'Engine_Speed','TI_Vehicle_Total_Engine_Dist',...
                    'Engine_Speed_200ms','Mass_Air_Flow*',
'Warning_Fault_Lamp')
            else
                if strcmp(engine(1:3),'ISX') == 1
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...
```

```
                    'Net_Engine_Torque', 'Engine_Speed',...
                    'IMD_Cyl_1_MisfirePercent','IMD_Cyl_2_MisfirePercent'
,...
                    'IMD_Cyl_3_MisfirePercent','IMD_Cyl_4_MisfirePercent',...
                    'IMD_Cyl_5_MisfirePercent','IMD_Cyl_6_MisfirePercent',...

'IMD_Active','IMD_Cyl_All_MisfirePercent','TI_Vehicle_Total_Engine_Dist',...
                    'Engine_Speed_200ms','Mass_Air_Flow*',
'Warning_Fault_Lamp')
                elseif strcmp(engine(1:3),'ISL') == 1
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...
                    'Net_Engine_Torque',
'Engine_Speed','TI_Vehicle_Total_Engine_Dist',...
                    'SBMD_IntMisfirePercentCyl1',
'SBMD_IntMisfirePercentCyl2', ...
                    'SBMD_IntMisfirePercentCyl3',
'SBMD_IntMisfirePercentCyl4', ...
                    'SBMD_IntMisfirePercentCyl5',
'SBMD_IntMisfirePercentCyl6',...
                    'SBMD_ContMisfirePercentCyl1',
'SBMD_ContMisfirePercentCyl2', ...
                    'SBMD_ContMisfirePercentCyl3',
'SBMD_ContMisfirePercentCyl4',...

'SBMD_ContMisfirePercentCyl5','SBMD_ContMisfirePercentCyl6',...
                    'Engine_Speed_Screen_1','Mass_Air_Flow*',
'Warning_Fault_Lamp')
                elseif strcmp(engine,'Badger') == 1 ||
strcmp(engine,'badger') == 1
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...

'OBD_Misfire_Cyl1_CAT_Ratio_200ms','OBD_Misfire_Cyl2_CAT_Ratio_200ms',...

'OBD_Misfire_Cyl3_CAT_Ratio_200ms','OBD_Misfire_Cyl4_CAT_Ratio_200ms',...

'OBD_Misfire_Cyl5_CAT_Ratio_200ms','OBD_Misfire_Cyl6_CAT_Ratio_200ms',...

'Net_Engine_Torque','Engine_Speed','TI_Vehicle_Total_Engine_Dist',...
                    'Engine_Speed_200ms','Mass_Air_Flow*',
'Warning_Fault_Lamp')
                end


            end
%***********If duty cycle plot is only
selected*****************************************%
```

```
        elseif (dutyDaily ==1 || duty ==1 || massduty == 1 || MAFdailyduty ==
1 || MAFmeshdaily ==1 || dailymesh==1) ...
                && (extraNorm==0 && normalV ==0 && normalVRange==0 && ...
                dailynormal == 0 && instantSVD==0 &&normalsvdtotaltime==0 &&
extractMilesNorm==0) ...
                && Dailymisfire == 0 && misfire == 0 && (svd==0 && wlamp==0
&& ...
                Dailysvd == 0 && extraSVD == 0 && scatterDaily ==0 &&
scatterSVD ==0 ...
                && scatterSVDdaily ==0 && avgSVDnoTor == 0 && maxEnv==0 &&
distSVD==0&& extDistSvd==0)
            load([fileLocations{x,2} '\' fileLocations{x,1}],
'Net_Engine_Torque',...
                'Engine_Speed','CBM_NetTorqueDemand_200ms',
'Engine_Speed_200ms',...
                'ENGN_Final_Torque_Cmd_Screen_1', 'Engine_Speed_Screen_1',...
                'ENGN_Final_Torque_Cmd_200ms',
'Mass_Air_Flow*','ENGN_Final_Torque_Cmd')
            if exist('CBM_NetTorqueDemand_200ms','var') == 1
                ENGN_Final_Torque_Cmd_200ms = CBM_NetTorqueDemand_200ms;
            elseif exist('ENGN_Final_Torque_Cmd_Screen_1','var')== 1
                ENGN_Final_Torque_Cmd_200ms =ENGN_Final_Torque_Cmd_Screen_1;
                Engine_Speed_200ms = Engine_Speed_Screen_1;
                Mass_Air_Flow_200ms = Mass_Air_Flow_Screen_1;
            elseif exist('ENGN_Final_Torque_Cmd','var') ==1
                Net_Engine_Torque =ENGN_Final_Torque_Cmd;
            end

%****************If warning lamp is only
selected*****************************************
        elseif wlamp==1 && (dutyDaily ==0 && duty ==0 && massduty == 0
&&MAFmeshdaily==0 && MAFdailyduty == 0)...
                && (extraNorm==0 && normalV ==0 && normalVRange==0 &&
dailynormal == 0 && instantSVD==0 &&normalsvdtotaltime==0) ...
                && Dailymisfire == 0 && misfire == 0 && (svd==0 && ...
                Dailysvd == 0 && extraSVD == 0 && scatterDaily ==0 &&
scatterSVD ==0 ...
                && scatterSVDdaily ==0 && avgSVDnoTor == 0 && maxEnv==0 &&
distSVD==0&& extDistSvd==0)
            load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                'Warning_Fault_Lamp')

%***************If svd and normalized voltage are both selected, will load
both***************%

        else
            if misfire ==0 && Dailymisfire == 0
                if strcmpi(engine(1:3),'ISX') == 1 ||
strcmpi(engine,'BADGER') == 1
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...
```

```
'Net_Engine_Torque','Engine_Speed','ECM_Run_Time_200ms',...
                    'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms' ,...
                    'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...
                    'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms',...
                    'CBM_NetTorqueDemand_200ms', 'Engine_Speed_200ms',...

'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms',...

'ENGN_Final_Torque_Cmd_200ms','TI_Vehicle_Total_Engine_Dist',...
                    'MTR_Active_200ms','Mass_Air_Flow*',
'Warning_Fault_Lamp')

                    if exist('CBM_NetTorqueDeman_200ms' ,'var') == 1 &&
exist('ENGN_Final_Torque_Cmd_200ms' ,'var') == 0
                        ENGN_Final_Torque_Cmd_200ms =
CBM_NetTorqueDeman_200ms;
                    end
                    if exist('ENGN_Final_Torque_Cmd_200ms', 'var') == 0 ||
exist('Engine_Speed_200ms', 'var') == 0
                        ENGN_Final_Torque_Cmd_200ms = Net_Engine_Torque;
                        Engine_Speed_200ms = Engine_Speed;
                    end
                    if exist('STC_Cyl_1_KnockOffset_200ms', 'var') == 0
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'STC_Cyl_1_Advance',...
                    'STC_Cyl_*', 'Intake_Manifold_Pressure')
                    end

                elseif strcmpi(engine(1:3),'ISL') == 1
                load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage' ,
'Warning_Fault_Lamp',...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...

'Net_Engine_Torque','Engine_Speed','ECM_Run_Time_200ms',...

'STC_Cyl_1_KnockOffset_Screen_1','STC_Cyl_2_KnockOffset_Screen_1',...

'STC_Cyl_3_KnockOffset_Screen_1','STC_Cyl_4_KnockOffset_Screen_1',...

'STC_Cyl_5_KnockOffset_Screen_1','STC_Cyl_6_KnockOffset_Screen_1',...
                    'SVD_Cyl_1_Voltage_Screen_1','SVD_Cyl_2_Voltage_Screen_1'
,...

'SVD_Cyl_3_Voltage_Screen_1','SVD_Cyl_4_Voltage_Screen_1',...

'SVD_Cyl_5_Voltage_Screen_1','SVD_Cyl_6_Voltage_Screen_1',...
```

```
                    'ENGN_Final_Torque_Cmd_Screen_1',
'Engine_Speed_Screen_1',...

'TI_Vehicle_Total_Engine_Dist','MTR_Active_200ms','Mass_Air_Flow*')

                try
                    SVD_Cyl_1_Voltage_200ms = SVD_Cyl_1_Voltage_Screen_1;
                    SVD_Cyl_2_Voltage_200ms = SVD_Cyl_2_Voltage_Screen_1;
                    SVD_Cyl_3_Voltage_200ms = SVD_Cyl_3_Voltage_Screen_1;
                    SVD_Cyl_4_Voltage_200ms = SVD_Cyl_4_Voltage_Screen_1;
                    SVD_Cyl_5_Voltage_200ms = SVD_Cyl_5_Voltage_Screen_1;
                    SVD_Cyl_6_Voltage_200ms = SVD_Cyl_6_Voltage_Screen_1;
                    ENGN_Final_Torque_Cmd_200ms =
ENGN_Final_Torque_Cmd_Screen_1;
                    Engine_Speed_200ms = Engine_Speed_Screen_1;
                    STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance_Screen_1;
                    Intake_Manifold_Pressure_200ms =
Intake_Manifold_Pressure_Screen_1;
                    STC_Cyl_1_KnockOffset_200ms =
STC_Cyl_1_KnockOffset_Screen_1;
                    STC_Cyl_2_KnockOffset_200ms =
STC_Cyl_2_KnockOffset_Screen_1;
                    STC_Cyl_3_KnockOffset_200ms =
STC_Cyl_3_KnockOffset_Screen_1;
                    STC_Cyl_4_KnockOffset_200ms =
STC_Cyl_4_KnockOffset_Screen_1;
                    STC_Cyl_5_KnockOffset_200ms =
STC_Cyl_5_KnockOffset_Screen_1;
                    STC_Cyl_6_KnockOffset_200ms =
STC_Cyl_6_KnockOffset_Screen_1;
                    Mass_Air_Flow_200ms = Mass_Air_Flow_Screen_1;

                 catch
                    STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance;
                    Intake_Manifold_Pressure_200ms =
Intake_Manifold_Pressure;
                    STC_Cyl_1_KnockOffset_200ms = STC_Cyl_1_KnockOffset;
                    STC_Cyl_2_KnockOffset_200ms = STC_Cyl_2_KnockOffset;
                    STC_Cyl_3_KnockOffset_200ms = STC_Cyl_3_KnockOffset;
                    STC_Cyl_4_KnockOffset_200ms = STC_Cyl_4_KnockOffset;
                    STC_Cyl_5_KnockOffset_200ms = STC_Cyl_5_KnockOffset;
                    STC_Cyl_6_KnockOffset_200ms = STC_Cyl_6_KnockOffset;
                    STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance;
                end
                end
            else
                if strcmpi(engine(1:3),'ISX') == 1
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...
                    'Net_Engine_Torque',
'Engine_Speed','TI_Vehicle_Total_Engine_Dist',...
```

```
                    'IMD_Cyl_1_MisfirePercent','IMD_Cyl_2_MisfirePercent'
,...
                    'IMD_Cyl_3_MisfirePercent','IMD_Cyl_4_MisfirePercent',...
                    'IMD_Cyl_5_MisfirePercent','IMD_Cyl_6_MisfirePercent',...

'IMD_Active','IMD_Cyl_All_MisfirePercent','ECM_Run_Time_200ms',...
                    'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms' ,...
                    'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...
                    'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms',...
                    'ENGN_Final_Torque_Cmd_200ms', 'Engine_Speed_200ms',...

'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms',...
                    'MTR_Active_200ms','Mass_Air_Flow*',
'Warning_Fault_Lamp')
                    if exist('STC_Cyl_1_KnockOffset_200ms', 'var') == 0
                        load([fileLocations{x,2} '\' fileLocations{x,1}],
'STC_Cyl_1_Advance',...
                        'STC_Cyl_*', 'Intake_Manifold_Pressure')

                    end
                    if exist('ENGN_Final_Torque_Cmd_200ms', 'var') == 0 ||
exist('Engine_Speed_200ms', 'var') == 0
                            ENGN_Final_Torque_Cmd_200ms = Net_Engine_Torque;
                            Engine_Speed_200ms = Engine_Speed;
                    end

                elseif strcmp(engine(1:3),'ISL') == 1
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...
                    'Net_Engine_Torque',
'Engine_Speed','ECM_Run_Time_200ms',...
                    'SVD_Cyl_1_Voltage_Screen_1','SVD_Cyl_2_Voltage_Screen_1'
,...

'SVD_Cyl_3_Voltage_Screen_1','SVD_Cyl_4_Voltage_Screen_1',...

'SVD_Cyl_5_Voltage_Screen_1','SVD_Cyl_6_Voltage_Screen_1',...
                    'SBMD_IntMisfirePercentCyl1',
'SBMD_IntMisfirePercentCyl2', ...
                    'SBMD_IntMisfirePercentCyl3',
'SBMD_IntMisfirePercentCyl4', ...
                    'SBMD_IntMisfirePercentCyl5',
'SBMD_IntMisfirePercentCyl6',...
                    'SBMD_ContMisfirePercentCyl1',
'SBMD_ContMisfirePercentCyl2', ...
```

```
                    'SBMD_ContMisfirePercentCyl3',
'SBMD_ContMisfirePercentCyl4',...

'SBMD_ContMisfirePercentCyl5','SBMD_ContMisfirePercentCyl6',...

'ENGN_Final_Torque_Cmd_Screen_1','Engine_Speed_Screen_1',...
                    'STC_Cyl*',
'Intake_Manifold_Pressure*','TI_Vehicle_Total_Engine_Dist',...
                    'MTR_Active*','Mass_Air_Flow*', 'Warning_Fault_Lamp')
                try
                    SVD_Cyl_1_Voltage_200ms = SVD_Cyl_1_Voltage_Screen_1;
                    SVD_Cyl_2_Voltage_200ms = SVD_Cyl_2_Voltage_Screen_1;
                    SVD_Cyl_3_Voltage_200ms = SVD_Cyl_3_Voltage_Screen_1;
                    SVD_Cyl_4_Voltage_200ms = SVD_Cyl_4_Voltage_Screen_1;
                    SVD_Cyl_5_Voltage_200ms = SVD_Cyl_5_Voltage_Screen_1;
                    SVD_Cyl_6_Voltage_200ms = SVD_Cyl_6_Voltage_Screen_1;
                    ENGN_Final_Torque_Cmd_200ms =
ENGN_Final_Torque_Cmd_Screen_1;
                    Engine_Speed_200ms = Engine_Speed_Screen_1;
                    STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance_Screen_1;
                    Intake_Manifold_Pressure_200ms =
Intake_Manifold_Pressure_Screen_1;
                    STC_Cyl_1_KnockOffset_200ms =
STC_Cyl_1_KnockOffset_Screen_1;
                    STC_Cyl_2_KnockOffset_200ms =
STC_Cyl_2_KnockOffset_Screen_1;
                    STC_Cyl_3_KnockOffset_200ms =
STC_Cyl_3_KnockOffset_Screen_1;
                    STC_Cyl_4_KnockOffset_200ms =
STC_Cyl_4_KnockOffset_Screen_1;
                    STC_Cyl_5_KnockOffset_200ms =
STC_Cyl_5_KnockOffset_Screen_1;
                    STC_Cyl_6_KnockOffset_200ms =
STC_Cyl_6_KnockOffset_Screen_1;
                    MTR_Active_200ms = MTR_Active_Screen_1;
                    Mass_Air_Flow_200ms = Mass_Air_Flow_Screen_1;

                catch
                    STC_Cyl_1_Advance_200ms = STC_Cyl_1_Advance;
                    Intake_Manifold_Pressure_200ms =
Intake_Manifold_Pressure;
                    STC_Cyl_1_KnockOffset_200ms = STC_Cyl_1_KnockOffset;
                    STC_Cyl_2_KnockOffset_200ms = STC_Cyl_2_KnockOffset;
                    STC_Cyl_3_KnockOffset_200ms = STC_Cyl_3_KnockOffset;
                    STC_Cyl_4_KnockOffset_200ms = STC_Cyl_4_KnockOffset;
                    STC_Cyl_5_KnockOffset_200ms = STC_Cyl_5_KnockOffset;
                    STC_Cyl_6_KnockOffset_200ms = STC_Cyl_6_KnockOffset;
                    MTR_Active_200ms = MTR_Active;

                end

                elseif strcmpi(engine,'BADGER') == 1
                    load([fileLocations{x,2} '\' fileLocations{x,1}],
'ECM_Run_Time',...
                    'SVD_Cyl_1_AverageVoltage','SVD_Cyl_2_AverageVoltage'
,...
```

```
                    'SVD_Cyl_3_AverageVoltage','SVD_Cyl_4_AverageVoltage',...
                    'SVD_Cyl_5_AverageVoltage','SVD_Cyl_6_AverageVoltage',...

'OBD_Misfire_Cyl1_CAT_Ratio_200ms','OBD_Misfire_Cyl2_CAT_Ratio_200ms',...

'OBD_Misfire_Cyl3_CAT_Ratio_200ms','OBD_Misfire_Cyl4_CAT_Ratio_200ms',...

'OBD_Misfire_Cyl5_CAT_Ratio_200ms','OBD_Misfire_Cyl6_CAT_Ratio_200ms',...

'Net_Engine_Torque','Engine_Speed','ECM_Run_Time_200ms',...
                    'SVD_Cyl_1_Voltage_200ms','SVD_Cyl_2_Voltage_200ms' ,...
                    'SVD_Cyl_3_Voltage_200ms','SVD_Cyl_4_Voltage_200ms',...

'SVD_Cyl_5_Voltage_200ms','SVD_Cyl_6_Voltage_200ms','MTR_Active_200ms',...
                    'CBM_NetTorqueDemand_200ms',
'Engine_Speed_200ms','TI_Vehicle_Total_Engine_Dist',...

'STC_Cyl_1_Advance_200ms','Intake_Manifold_Pressure_200ms',...

'STC_Cyl_1_KnockOffset_200ms','STC_Cyl_2_KnockOffset_200ms'...

,'STC_Cyl_3_KnockOffset_200ms','STC_Cyl_4_KnockOffset_200ms',...

'STC_Cyl_5_KnockOffset_200ms','STC_Cyl_6_KnockOffset_200ms',...
                    'Mass_Air_Flow*', 'Warning_Fault_Lamp')
                    ENGN_Final_Torque_Cmd_200ms = CBM_NetTorqueDemand_200ms;
                end
            end
        end
    catch
        fprintf(['\n' fileLocations{x,1} ' does not exist\n']);
        warning('Problem loading file');
        continue;
    end

    fprintf(['Processisng ' fileLocations{x,1} '\n']);

    if wlamp == 1
        try
            if any(Warning_Fault_Lamp) == 1
                gf = figure;
                set(gf,'Name',['Warning Fault Lamp ' fileLocations{x,1}]);
                time = (ECM_Run_Time-min(ECM_Run_Time))/3600;
                plot(time, Warning_Fault_Lamp);
                title_(fileLocations{x,1});
                xlabel('ECM Run Time (hours)'); ylabel('Warning Fault Lamp');
                grid on
            else
                fprintf(['\n Warning Fault Lamp did not turn on for '
fileLocations{x,1}]);
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting warning fault lamp');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
```

```
    end

    if scatterDaily == 1 %scatter plots daily torque vs speed
        try
            figure;
            scatter(Engine_Speed,Net_Engine_Torque);
            ylim([-2500 2500]); ylabel('Torque (Nm)'); xlabel('Engine Speed
(RPM)');
            title_(fileLocations{x,1});
            grid on
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting Scatter Plots of Torque vs
Speed');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if dutyDaily == 1 %daily duty cycle plots
        try
            if exist('ENGN_Final_Torque_Cmd_200ms','var')==1 && ...
                    exist('Engine_Speed_200ms','var')==1
                dutycycle(fileLocations{x,1},...
                    ENGN_Final_Torque_Cmd_200ms,
Engine_Speed_200ms,fileLocations{x,1},1);
            else
                dutycycle(fileLocations{x,1},...
                    Net_Engine_Torque, Engine_Speed,fileLocations{x,1},1);
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting Daily Duty Cycle Plot');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end
    if dailymesh == 1 %daily mesh plots of torque and speed
        try
            if exist('ENGN_Final_Torque_Cmd_200ms','var')==1 && ...
                    exist('Engine_Speed_200ms','var')==1
                dutycycle(fileLocations{x,1},...
                    ENGN_Final_Torque_Cmd_200ms,
Engine_Speed_200ms,fileLocations{x,1},2);
            else
                dutycycle(fileLocations{x,1},...
                    Net_Engine_Torque, Engine_Speed,fileLocations{x,1},2);
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting Daily Duty Cycle Plot');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if MAFdailyduty == 1 %daily MAF duty cycle plots
        try
           if exist('Mass_Air_Flow_200ms','var')==1 && ...
```

```
                    exist('Engine_Speed_200ms','var')==1
                MAFdutycycle(fileLocations{x,1},...
                    Mass_Air_Flow_200ms,
Engine_Speed_200ms,fileLocations{x,1},1);
            else
                MAFdutycycle(fileLocations{x,1},...
                Mass_Air_Flow, Engine_Speed,fileLocations{x,1},1);
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting MAF Daily Duty Cycle Plot');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end
    if MAFmeshdaily == 1 %daily MAF mesh plots
        try

           if exist('Mass_Air_Flow_200ms','var')==1 && ...
                    exist('Engine_Speed_200ms','var')==1
            MAFdutycycle(fileLocations{x,1},...
                Mass_Air_Flow_200ms,
Engine_Speed_200ms,fileLocations{x,1},2);
            else
                MAFdutycycle(fileLocations{x,1},...
                Mass_Air_Flow, Engine_Speed,fileLocations{x,1},2);
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting Daily MAF mesh Plot');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if duty == 1  %combined duty plot
        try
            if exist('ENGN_Final_Torque_Cmd_200ms','var')==1 && ...
                    exist('Engine_Speed_200ms','var')==1
                dTotalTorque = [dTotalTorque; ENGN_Final_Torque_Cmd_200ms];
                dTotalSpeed = [dTotalSpeed; Engine_Speed_200ms];
            else
                dTotalTorque = [dTotalTorque; Net_Engine_Torque];
                dTotalSpeed = [dTotalSpeed; Engine_Speed];
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in saving variables for Duty Cycle Plot');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end
     if massduty == 1 %combined mass air flow duty plot
        try
            if exist('Mass_Air_Flow_200ms','var')==1 && ...
                    exist('Engine_Speed_200ms','var')==1
                mTotalSpeed = [mTotalSpeed; Engine_Speed_200ms];
                mTotalMAF = [ mTotalMAF; Mass_Air_Flow_200ms];
            else
```

```
                mTotalSpeed = [dTotalSpeed; Engine_Speed];
                mTotalMAF = [mTotalMAF; Mass_Air_Flow_200ms];
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in saving variables for Duty Cycle Plot');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if scatterSVDdaily == 1 %scatter plots daily svd into voltage ranges
        try
            plotTorque_Speed(fileLocations{x,1},
SVD_Cyl_1_AverageVoltage,SVD_Cyl_2_AverageVoltage ,...
           SVD_Cyl_3_AverageVoltage,SVD_Cyl_4_AverageVoltage,...
           SVD_Cyl_5_AverageVoltage,SVD_Cyl_6_AverageVoltage,...
           Net_Engine_Torque, Engine_Speed);
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting Scatter Plots of SVD');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if scatterSVD == 1 %scatter plots daily svd, V ranges combined
        try
            PlotSVDTorque_Speed_combined(fileLocations{x,1},
SVD_Cyl_1_AverageVoltage, SVD_Cyl_2_AverageVoltage,...
           SVD_Cyl_3_AverageVoltage,SVD_Cyl_4_AverageVoltage,...
           SVD_Cyl_5_AverageVoltage,SVD_Cyl_6_AverageVoltage,...
           Net_Engine_Torque, Engine_Speed);

        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in Plotting Scatter Plots of SVD');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if svd == 1 || extraSVD == 1 || maxEnv == 1 || avgSVDnoTor == 1 ...
            || distSVD == 1 || extDistSvd == 1
        %catch spark plug voltages
        TotalInterval_SVD_Cyl_1_Voltage =
[TotalInterval_SVD_Cyl_1_Voltage;SVD_Cyl_1_AverageVoltage];
        TotalInterval_SVD_Cyl_2_Voltage =
[TotalInterval_SVD_Cyl_2_Voltage;SVD_Cyl_2_AverageVoltage];
        TotalInterval_SVD_Cyl_3_Voltage =
[TotalInterval_SVD_Cyl_3_Voltage;SVD_Cyl_3_AverageVoltage];
        TotalInterval_SVD_Cyl_4_Voltage =
[TotalInterval_SVD_Cyl_4_Voltage;SVD_Cyl_4_AverageVoltage];
        TotalInterval_SVD_Cyl_5_Voltage =
[TotalInterval_SVD_Cyl_5_Voltage;SVD_Cyl_5_AverageVoltage];
        TotalInterval_SVD_Cyl_6_Voltage =
[TotalInterval_SVD_Cyl_6_Voltage;SVD_Cyl_6_AverageVoltage];
        TotalTorque = [TotalTorque; Net_Engine_Torque];
        Totalspeed = [Totalspeed ; Engine_Speed];
```

```
            if distSVD == 1 || extDistSvd == 1
                totalmeters = [totalmeters; TI_Vehicle_Total_Engine_Dist];
            end
        ECM_Run_Time(~any(~isnan(ECM_Run_Time), 2),:)=[];
%         if isempty(ECM_Run_Time) == 1
%             fprintf(['\n' fileLocations{x,1} ' has NO data \n']);
%             continue;
%         end %IF THIS WORKS DELETE IT
        if length(total_run_time) ~= 1
            todayranfor = max(total_run_time)+((ECM_Run_Time -
min(ECM_Run_Time))/3600); %in hrs
            total_run_time = [total_run_time; todayranfor];
        else
            todayranfor =((ECM_Run_Time - ECM_Run_Time(1))/3600);
            total_run_time = todayranfor;
        end

%         todayranfor = ((ECM_Run_Time(end) - ECM_Run_Time(1))/3600); %in hrs
%         total_run_time =  total_run_time + todayranfor;

%         if maxEnv == 1 %delete
%             combinedRunTime = [combinedRunTime ; ECM_Run_Time];
%         end
    end

    if extraNorm == 1 || instantSVD == 1 || normalsvdtotaltime == 1 ||
extractMilesNorm==1
        NTotalInterval_SVD_Cyl_1_Voltage =
[NTotalInterval_SVD_Cyl_1_Voltage;SVD_Cyl_1_Voltage_200ms];
        NTotalInterval_SVD_Cyl_2_Voltage =
[NTotalInterval_SVD_Cyl_2_Voltage;SVD_Cyl_2_Voltage_200ms];
        NTotalInterval_SVD_Cyl_3_Voltage =
[NTotalInterval_SVD_Cyl_3_Voltage;SVD_Cyl_3_Voltage_200ms];
        NTotalInterval_SVD_Cyl_4_Voltage =
[NTotalInterval_SVD_Cyl_4_Voltage;SVD_Cyl_4_Voltage_200ms];
        NTotalInterval_SVD_Cyl_5_Voltage =
[NTotalInterval_SVD_Cyl_5_Voltage;SVD_Cyl_5_Voltage_200ms];
        NTotalInterval_SVD_Cyl_6_Voltage =
[NTotalInterval_SVD_Cyl_6_Voltage;SVD_Cyl_6_Voltage_200ms];
        NTotalTorque = [NTotalTorque; ENGN_Final_Torque_Cmd_200ms];
        NTotalspeed = [NTotalspeed ; Engine_Speed_200ms];
        ECM_Run_Time_200ms(~any(~isnan(ECM_Run_Time_200ms), 2),:)=[];
%         if isempty(ECM_Run_Time_200ms) == 1
%             fprintf(['\n' fileLocations{x,1} ' has NO data \n']);
%             continue;
%         end DELETE
        if length(Ntotal_run_time) ~= 1
            Ntodayranfor = max(Ntotal_run_time)+((ECM_Run_Time_200ms -
ECM_Run_Time_200ms(1))/3600); %in hrs
            Ntotal_run_time = [Ntotal_run_time; Ntodayranfor];
        else
            Ntodayranfor =((ECM_Run_Time_200ms -
ECM_Run_Time_200ms(1))/3600);
            Ntotal_run_time = Ntodayranfor;
        end
```

```
        if extraNorm == 1 || normalsvdtotaltime == 1 || extractMilesNorm==1

TotalSTC_Cyl_1_Advance=[TotalSTC_Cyl_1_Advance;STC_Cyl_1_Advance_200ms ];

TotalIntake_Manifold_Pressure=[TotalIntake_Manifold_Pressure;Intake_Manifold_
Pressure_200ms ];
            TotalSTC_Cyl_1_Knock = [TotalSTC_Cyl_1_Knock
;STC_Cyl_1_KnockOffset_200ms];
            TotalSTC_Cyl_2_Knock=[TotalSTC_Cyl_2_Knock;
STC_Cyl_2_KnockOffset_200ms];
            TotalSTC_Cyl_3_Knock=[
TotalSTC_Cyl_3_Knock;STC_Cyl_3_KnockOffset_200ms];
            TotalSTC_Cyl_4_Knock=[
TotalSTC_Cyl_4_Knock;STC_Cyl_4_KnockOffset_200ms];
            TotalSTC_Cyl_5_Knock=[
TotalSTC_Cyl_5_Knock;STC_Cyl_5_KnockOffset_200ms];
            TotalSTC_Cyl_6_Knock=[
TotalSTC_Cyl_6_Knock;STC_Cyl_6_KnockOffset_200ms];
            NTotalmotoring = [NTotalmotoring; MTR_Active_200ms];
            if extractMilesNorm ==1
                NTotalmeters = [NTotalmeters; TI_Vehicle_Total_Engine_Dist];
            end
        end
    end


    if Dailysvd == 1
        try
            PlotDAILYsvd_Time(fileLocations{x,1},
SVD_Cyl_1_AverageVoltage,SVD_Cyl_2_AverageVoltage ,...
            SVD_Cyl_3_AverageVoltage,SVD_Cyl_4_AverageVoltage,...
            SVD_Cyl_5_AverageVoltage,SVD_Cyl_6_AverageVoltage,...
            Net_Engine_Torque, Engine_Speed, ECM_Run_Time);

        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in plotting daily SVD');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if Dailymisfire == 1
        try
            if strcmp(engine(1:3),'ISX') == 1
                PlotDailyMisfires(fileLocations{x,1}, engine,
ECM_Run_Time,...
                        Net_Engine_Torque, Engine_Speed,...
                        IMD_Cyl_1_MisfirePercent, IMD_Cyl_2_MisfirePercent
,...
                        IMD_Cyl_3_MisfirePercent,
IMD_Cyl_4_MisfirePercent,...
                        IMD_Cyl_5_MisfirePercent,
IMD_Cyl_6_MisfirePercent,...
                        IMD_Active,IMD_Cyl_All_MisfirePercent);
```

```
            elseif strcmp(engine,'Badger') == 1 || strcmp(engine,'badger')==1
                 if exist('OBD_Misfire_Cyl1_CAT_Ratio_200ms', 'var') == 1
                     PlotDailyMisfires(fileLocations{x,1},
engine,ECM_Run_Time,...
                         Net_Engine_Torque, Engine_Speed,...

OBD_Misfire_Cyl1_CAT_Ratio_200ms,OBD_Misfire_Cyl2_CAT_Ratio_200ms,...

OBD_Misfire_Cyl3_CAT_Ratio_200ms,OBD_Misfire_Cyl4_CAT_Ratio_200ms,...

OBD_Misfire_Cyl5_CAT_Ratio_200ms,OBD_Misfire_Cyl6_CAT_Ratio_200ms)
                 else
                     fprintf(['\nFILE: ' fileLocations{x,1} ' does NOT
contain OBD misfire data\n']);
                 end

            elseif strcmp(engine(1:3),'ISL') == 1
                PlotDailyMisfires(fileLocations{x,1}, engine,ECM_Run_Time,...
                        Net_Engine_Torque, Engine_Speed,...
                        SBMD_IntMisfirePercentCyl1,
SBMD_IntMisfirePercentCyl2, ...
                        SBMD_IntMisfirePercentCyl3,
SBMD_IntMisfirePercentCyl4, ...
                        SBMD_IntMisfirePercentCyl5,
SBMD_IntMisfirePercentCyl6,1)

                 PlotDailyMisfires(fileLocations{x,1},
engine,ECM_Run_Time,...
                        Net_Engine_Torque, Engine_Speed,...
                        SBMD_ContMisfirePercentCyl1,
SBMD_ContMisfirePercentCyl2, ...
                        SBMD_ContMisfirePercentCyl3,
SBMD_ContMisfirePercentCyl4,...
                        SBMD_ContMisfirePercentCyl5,
SBMD_ContMisfirePercentCyl6,2)
            end
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in plotting daily Misfires');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if misfire == 1
        try
            Mrun_time =  [Mrun_time; ECM_Run_Time];

             if strcmp(engine(1:3),'ISX') == 1
                total_Misfire_1 = [total_Misfire_1;
IMD_Cyl_1_MisfirePercent];
                total_Misfire_2 = [total_Misfire_2;
IMD_Cyl_2_MisfirePercent];
                total_Misfire_3 = [total_Misfire_3;
IMD_Cyl_3_MisfirePercent];
                total_Misfire_4 = [total_Misfire_4;
IMD_Cyl_4_MisfirePercent];
```

```
                total_Misfire_5 = [total_Misfire_5;
IMD_Cyl_5_MisfirePercent];
                total_Misfire_6 = [total_Misfire_6;
IMD_Cyl_6_MisfirePercent];
                total_IMDALL = [total_IMDALL; IMD_Cyl_All_MisfirePercent];
                total_MisfireActive = [total_MisfireActive; IMD_Active];
                MTotalTorque = [MTotalTorque; Net_Engine_Torque];
                MTotalspeed = [MTotalspeed; Engine_Speed];

            elseif strcmp(engine,'Badger') == 1 ||
strcmp(engine,'badger')==1
                if exist('OBD_Misfire_Cyl1_CAT_Ratio_200ms', 'var') == 1
                    total_Misfire_1 = [total_Misfire_1;
OBD_Misfire_Cyl1_CAT_Ratio_200ms];
                    total_Misfire_2 = [total_Misfire_2;
OBD_Misfire_Cyl2_CAT_Ratio_200ms];
                    total_Misfire_3 = [total_Misfire_3;
OBD_Misfire_Cyl3_CAT_Ratio_200ms];
                    total_Misfire_4 = [total_Misfire_4;
OBD_Misfire_Cyl4_CAT_Ratio_200ms];
                    total_Misfire_5 = [total_Misfire_5;
OBD_Misfire_Cyl5_CAT_Ratio_200ms];
                    total_Misfire_6 = [total_Misfire_6;
OBD_Misfire_Cyl6_CAT_Ratio_200ms];
                    MTotalTorque = [MTotalTorque; Net_Engine_Torque];
                    MTotalspeed = [MTotalspeed; Engine_Speed];
                 else
                     fprintf(['\nFILE: ' fileLocations{x,1} ' does NOT
contain OBD misfire data\n']);
                end
            elseif strcmp(engine(1:3),'ISL') == 1
                total_Misfire_1 = [total_Misfire_1;
SBMD_IntMisfirePercentCyl1];
                total_Misfire_2 = [total_Misfire_2;
SBMD_IntMisfirePercentCyl2];
                total_Misfire_3 = [total_Misfire_3;
SBMD_IntMisfirePercentCyl3];
                total_Misfire_4 = [total_Misfire_4;
SBMD_IntMisfirePercentCyl4];
                total_Misfire_5 = [total_Misfire_5;
SBMD_IntMisfirePercentCyl5];
                total_Misfire_6 = [total_Misfire_6;
SBMD_IntMisfirePercentCyl6];

                totalMisfire_1 = [totalMisfire_1;
SBMD_ContMisfirePercentCyl1];
                totalMisfire_2 = [totalMisfire_2;
SBMD_ContMisfirePercentCyl2];
                totalMisfire_3 = [totalMisfire_3;
SBMD_ContMisfirePercentCyl3];
                totalMisfire_4 = [totalMisfire_4;
SBMD_ContMisfirePercentCyl4];
                totalMisfire_5 = [totalMisfire_5;
SBMD_ContMisfirePercentCyl5];
                totalMisfire_6 = [totalMisfire_6;
SBMD_ContMisfirePercentCyl6];
```

```
                MTotalTorque = [MTotalTorque; Net_Engine_Torque];
                MTotalspeed = [MTotalspeed; Engine_Speed];

            end

        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in plotting daily Misfires');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if normalV == 1
        try

normalizeSVDVoltage(STC_Cyl_1_Advance_200ms,Intake_Manifold_Pressure_200ms,..
.
            STC_Cyl_1_KnockOffset_200ms,STC_Cyl_2_KnockOffset_200ms,...
            STC_Cyl_3_KnockOffset_200ms,STC_Cyl_4_KnockOffset_200ms,...
            STC_Cyl_5_KnockOffset_200ms,STC_Cyl_6_KnockOffset_200ms,...
           fileLocations{x,1},
SVD_Cyl_1_Voltage_200ms,SVD_Cyl_2_Voltage_200ms ,...
           SVD_Cyl_3_Voltage_200ms,SVD_Cyl_4_Voltage_200ms,...
           SVD_Cyl_5_Voltage_200ms,SVD_Cyl_6_Voltage_200ms,...
           ENGN_Final_Torque_Cmd_200ms, Engine_Speed_200ms, engine,...
           MTR_Active_200ms);
       %if knockOffset200ms doesnt exist run this below, the average
%
normalizeSVDVoltage(STC_Cyl_1_Advance_200ms,Intake_Manifold_Pressure_200ms,..
.
%             STC_Cyl_1_KnockOffset_200ms,STC_Cyl_2_KnockOffset_200ms,...
%             STC_Cyl_3_KnockOffset_200ms,STC_Cyl_4_KnockOffset_200ms,...
%             STC_Cyl_5_KnockOffset_200ms,STC_Cyl_6_KnockOffset_200ms,...
%            fileLocations{x,1},
SVD_Cyl_1_AverageVoltage,SVD_Cyl_2_AverageVoltage,...
%            SVD_Cyl_3_AverageVoltage,SVD_Cyl_4_AverageVoltage,...
%            SVD_Cyl_5_AverageVoltage,SVD_Cyl_6_AverageVoltage,...
%            ENGN_Final_Torque_Cmd_200ms, Engine_Speed_200ms, engine)


        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in scatter plotting normalized voltages');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if normalVRange == 1
        try
%
normalizeSVDVoltageRanges(STC_Cyl_1_Advance_200ms,Intake_Manifold_Pressure_20
0ms,...
%                STC_Cyl_1_KnockOffset_200ms,STC_Cyl_2_KnockOffset_200ms,...
%                STC_Cyl_3_KnockOffset_200ms,STC_Cyl_4_KnockOffset_200ms,...
%                STC_Cyl_5_KnockOffset_200ms,STC_Cyl_6_KnockOffset_200ms,...
```

```
%                 fileLocations{x,1},
SVD_Cyl_1_Voltage_200ms,SVD_Cyl_2_Voltage_200ms ,...
%                 SVD_Cyl_3_Voltage_200ms,SVD_Cyl_4_Voltage_200ms,...
%                 SVD_Cyl_5_Voltage_200ms,SVD_Cyl_6_Voltage_200ms,...
%                 ENGN_Final_Torque_Cmd_200ms, Engine_Speed_200ms, engine)

normalizeSVDVoltageRanges(STC_Cyl_1_Advance_200ms,Intake_Manifold_Pressure_20
0ms,...
                STC_Cyl_1_KnockOffset_200ms,STC_Cyl_2_KnockOffset_200ms,...
                STC_Cyl_3_KnockOffset_200ms,STC_Cyl_4_KnockOffset_200ms,...
                STC_Cyl_5_KnockOffset_200ms,STC_Cyl_6_KnockOffset_200ms,...
                fileLocations{x,1},
SVD_Cyl_1_Voltage_200ms,SVD_Cyl_2_Voltage_200ms ,...
                SVD_Cyl_3_Voltage_200ms,SVD_Cyl_4_Voltage_200ms,...
                SVD_Cyl_5_Voltage_200ms,SVD_Cyl_6_Voltage_200ms,...
                ENGN_Final_Torque_Cmd_200ms, Engine_Speed_200ms,...
                engine, MTR_Active_200ms)
        catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in scatter plotting normalized voltage
ranges');
            set(handles.PLOTpushbutton,'string', {'Error'});
        end
    end

    if dailynormal == 1
        try
            if exist('STC_Cyl_1_KnockOffset_200ms' ,'var')== 1

normalVSdailytime(STC_Cyl_1_Advance_200ms,Intake_Manifold_Pressure_200ms,...
                STC_Cyl_1_KnockOffset_200ms,STC_Cyl_2_KnockOffset_200ms,...
                STC_Cyl_3_KnockOffset_200ms,STC_Cyl_4_KnockOffset_200ms,...
                STC_Cyl_5_KnockOffset_200ms,STC_Cyl_6_KnockOffset_200ms,...
                fileLocations{x,1},
SVD_Cyl_1_Voltage_200ms,SVD_Cyl_2_Voltage_200ms ,...
                SVD_Cyl_3_Voltage_200ms,SVD_Cyl_4_Voltage_200ms,...
                SVD_Cyl_5_Voltage_200ms,SVD_Cyl_6_Voltage_200ms,engine,...
                ENGN_Final_Torque_Cmd_200ms, Engine_Speed_200ms, ...
                ECM_Run_Time_200ms,MTR_Active_200ms);
            else
            normalVSdailytime(STC_Cyl_1_Advance,Intake_Manifold_Pressure,...
            STC_Cyl_1_KnockOffset,STC_Cyl_2_KnockOffset,...
            STC_Cyl_3_KnockOffset,STC_Cyl_4_KnockOffset,...
            STC_Cyl_5_KnockOffset,STC_Cyl_6_KnockOffset,...
           fileLocations{x,1},
SVD_Cyl_1_AverageVoltage,SVD_Cyl_2_AverageVoltage,...
           SVD_Cyl_3_AverageVoltage,SVD_Cyl_4_AverageVoltage,...
           SVD_Cyl_5_AverageVoltage,SVD_Cyl_6_AverageVoltage,...
           ENGN_Final_Torque_Cmd_200ms, Engine_Speed_200ms,...
           ECM_Run_Time_200ms,MTR_Active)
            end

       catch
            set(handles.figure1, 'pointer', 'arrow');
            warning('Error Occured in plotting normalized voltage vs time');
            set(handles.PLOTpushbutton,'string', {'Error'});
```

```
        end
    end

    %%% to add daily plots add it here (before the end) + add data for
    %%% combined plots

%     clearvars('*', '-except', 'Total*', 'total*',
'file*','NTot*','Ntota*','engine','MTot*','Mtot*',...
%         'Mrun*', 'gpath', 'path', 'svd','daily*','norm*','extra*',
'instant*','Daily*','fleets','Input',...
%         'dTotalSpeed','mTotalMAF', 'scatter*' , 'avgSVDnoTor', 'maxEnv',
'distSVD', 'extDistSvd','wlamp',...
%         'savegif', 'save', 'mis*', 'duty*', 'mass*', 'MAF*',
'it','event*','handles','hOb*','x')
%
    end  %**************end of for loop for one unit **went through all the
dates/files***

    if exist( 'svd','var') == 1
        if svd == 1
            try
            PlotSvd_Time(fileLocations{1,1},
TotalInterval_SVD_Cyl_1_Voltage,...

TotalInterval_SVD_Cyl_2_Voltage,TotalInterval_SVD_Cyl_3_Voltage,...
            TotalInterval_SVD_Cyl_4_Voltage,
TotalInterval_SVD_Cyl_5_Voltage,...
            TotalInterval_SVD_Cyl_6_Voltage, TotalTorque,...
            Totalspeed, total_run_time, 0, fileLocations{end,1});
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting SVD for entire time
interval');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
    end

    if exist( 'avgSVDnoTor','var') == 1
        if avgSVDnoTor == 1
            try
            PlotSvd_Time(fileLocations{1,1},
TotalInterval_SVD_Cyl_1_Voltage,...

TotalInterval_SVD_Cyl_2_Voltage,TotalInterval_SVD_Cyl_3_Voltage,...
            TotalInterval_SVD_Cyl_4_Voltage,
TotalInterval_SVD_Cyl_5_Voltage,...
            TotalInterval_SVD_Cyl_6_Voltage, TotalTorque,...
            Totalspeed, total_run_time, 1, fileLocations{end,1});
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting SVD for entire time
interval');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
```

```
    end

    if exist( 'maxEnv','var') == 1
        if maxEnv == 1
            plotmaxEnvelope(
fileLocations{1,1},TotalInterval_SVD_Cyl_1_Voltage,...

TotalInterval_SVD_Cyl_2_Voltage,TotalInterval_SVD_Cyl_3_Voltage,...
            TotalInterval_SVD_Cyl_4_Voltage,
TotalInterval_SVD_Cyl_5_Voltage,...
            TotalInterval_SVD_Cyl_6_Voltage,total_run_time,
fileLocations{end,1});
        end
    end

    if exist( 'extraSVD','var') == 1
        if extraSVD == 1
            XY = str2double(get(handles.extrapSVDtoXYhours, 'String'));

extrapolateSVD(fileLocations{1,1},TotalInterval_SVD_Cyl_1_Voltage,...
            TotalInterval_SVD_Cyl_2_Voltage,
TotalInterval_SVD_Cyl_3_Voltage,...
            TotalInterval_SVD_Cyl_4_Voltage,
TotalInterval_SVD_Cyl_5_Voltage,...
            TotalInterval_SVD_Cyl_6_Voltage, ...
            total_run_time, XY, fileLocations{end,1});
        end
    end

     if exist( 'instantSVD','var') == 1
        if instantSVD == 1
            try
            PlotSvd_Time(fileLocations{1,1},
NTotalInterval_SVD_Cyl_1_Voltage,...

NTotalInterval_SVD_Cyl_2_Voltage,NTotalInterval_SVD_Cyl_3_Voltage,...
            NTotalInterval_SVD_Cyl_4_Voltage,
NTotalInterval_SVD_Cyl_5_Voltage,...
            NTotalInterval_SVD_Cyl_6_Voltage, NTotalTorque,...
            NTotalspeed, Ntotal_run_time, 2, fileLocations{end,1});
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting SVD for entire time
interval');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
     end

     if exist( 'duty','var') == 1 %plots duty cycle
        if duty == 1
            try
              dutycycle(fileLocations{1,1}, dTotalTorque,...
                dTotalSpeed, fileLocations{end,1},0); %0 indicates type of
plot
            catch
```

```
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting duty plot for entire time
interval');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
    end

    if exist( 'massduty','var') == 1 %plots mass air flow duty plots
        if massduty == 1
            try
              MAFdutycycle(fileLocations{1,1}, mTotalMAF,...
                mTotalSpeed, fileLocations{end,1},0); %0 indicates type of
plot
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting duty plot for entire time
interval');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
    end

    if exist( 'distSVD','var') == 1 %plots svd vs miles
        if distSVD == 1
            try

PlotMiles_SVD(fileLocations{1,1},fileLocations{end,1},totalmeters, ...
                TotalInterval_SVD_Cyl_1_Voltage, Totalspeed,...

TotalInterval_SVD_Cyl_2_Voltage,TotalInterval_SVD_Cyl_3_Voltage,...
                TotalInterval_SVD_Cyl_4_Voltage,
TotalInterval_SVD_Cyl_5_Voltage,...
                TotalInterval_SVD_Cyl_6_Voltage, TotalTorque)
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting SVD vs Miles');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
    end

   if exist( 'extractMilesNorm','var') == 1 %plots svd vs miles
        if extractMilesNorm == 1
            try
             xymiles = str2double(get(handles.xmilesNormedit, 'String'));

extrapolateNormtoMiles(fileLocations{1,1},fileLocations{end,1},NTotalmeters,
...
                NTotalInterval_SVD_Cyl_1_Voltage, ...

NTotalInterval_SVD_Cyl_2_Voltage,NTotalInterval_SVD_Cyl_3_Voltage,...
                NTotalInterval_SVD_Cyl_4_Voltage,
NTotalInterval_SVD_Cyl_5_Voltage,...
                NTotalInterval_SVD_Cyl_6_Voltage, xymiles,NTotalmotoring,...
                NTotalTorque,NTotalspeed, engine,...
```

```
                TotalSTC_Cyl_1_Advance,TotalIntake_Manifold_Pressure,...
                TotalSTC_Cyl_1_Knock,TotalSTC_Cyl_2_Knock,...
                TotalSTC_Cyl_3_Knock,TotalSTC_Cyl_4_Knock,...
                TotalSTC_Cyl_5_Knock,TotalSTC_Cyl_6_Knock)
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in Extrapolating Target Miles to
Normalized Voltage');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
    end

    if exist( 'extDistSvd','var') == 1 %plots svd vs miles
        if extDistSvd == 1
            try
             xymiles = str2double(get(handles.Extractmilesedit, 'String'));

extrapolateSVDtoMiles(fileLocations{1,1},fileLocations{end,1},totalmeters,
...
                TotalInterval_SVD_Cyl_1_Voltage, ...

TotalInterval_SVD_Cyl_2_Voltage,TotalInterval_SVD_Cyl_3_Voltage,...
                TotalInterval_SVD_Cyl_4_Voltage,
TotalInterval_SVD_Cyl_5_Voltage,...
                TotalInterval_SVD_Cyl_6_Voltage, xymiles)
            catch
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting SVD vs Miles');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end
    end

    if exist( 'extraNorm','var') == 1 %extrapolate normalized V to xhours
        if extraNorm == 1
            xHours = str2double(get(handles.editNormhours, 'String'));

extrapolateNormalized(TotalSTC_Cyl_1_Advance,TotalIntake_Manifold_Pressure,..
.
            TotalSTC_Cyl_1_Knock,TotalSTC_Cyl_2_Knock,...
            TotalSTC_Cyl_3_Knock,TotalSTC_Cyl_4_Knock,...
            TotalSTC_Cyl_5_Knock,TotalSTC_Cyl_6_Knock,...
            fileLocations{1,1},NTotalInterval_SVD_Cyl_1_Voltage,...
            NTotalInterval_SVD_Cyl_2_Voltage,
NTotalInterval_SVD_Cyl_3_Voltage,...
            NTotalInterval_SVD_Cyl_4_Voltage,
NTotalInterval_SVD_Cyl_5_Voltage,...
            NTotalInterval_SVD_Cyl_6_Voltage, ...
            Ntotal_run_time, xHours, engine,
fileLocations{end,1},NTotalmotoring,...
            NTotalTorque, NTotalspeed)
        end
    end
```

```
    if exist( 'normalsvdtotaltime','var') == 1 %plots normalized V for total
time
        if normalsvdtotaltime == 1

plotTotalNormalized(TotalSTC_Cyl_1_Advance,TotalIntake_Manifold_Pressure,...
            TotalSTC_Cyl_1_Knock,TotalSTC_Cyl_2_Knock,...
            TotalSTC_Cyl_3_Knock,TotalSTC_Cyl_4_Knock,...
            TotalSTC_Cyl_5_Knock,TotalSTC_Cyl_6_Knock,...
            fileLocations{1,1},NTotalInterval_SVD_Cyl_1_Voltage,...
            NTotalInterval_SVD_Cyl_2_Voltage,
NTotalInterval_SVD_Cyl_3_Voltage,...
            NTotalInterval_SVD_Cyl_4_Voltage,
NTotalInterval_SVD_Cyl_5_Voltage,...
            NTotalInterval_SVD_Cyl_6_Voltage, fileLocations{end,1},...
            Ntotal_run_time, engine, NTotalTorque,
NTotalspeed,NTotalmotoring)
        end
    end

    if exist( 'misfire','var') == 1
        if misfire == 1
            try
            if strcmp(engine(1:3),'ISX') == 1
                PlotDailyMisfires(fileLocations{x,1}, engine, Mrun_time,...
                        MTotalTorque, MTotalspeed,...
                        total_Misfire_1, total_Misfire_2,...
                        total_Misfire_3, total_Misfire_4,...
                        total_Misfire_5, total_Misfire_6,...
                        total_MisfireActive, total_IMDALL);

            elseif strcmp(engine,'Badger') == 1 || strcmp(engine,'badger')==1
                 if exist('OBD_Misfire_Cyl1_CAT_Ratio_200ms', 'var') == 1
                 PlotDailyMisfires(fileLocations{x,1}, engine, Mrun_time,...
                         MTotalTorque, MTotalspeed,...
                         total_Misfire_1, total_Misfire_2,...
                         total_Misfire_3, total_Misfire_4,...
                         total_Misfire_5, total_Misfire_6);
                 else
                     fprintf(['\nFILE: ' fileLocations{x,1} ' does NOT
contain OBD misfire data\n']);
                 end

            elseif strcmp(engine(1:3),'ISL') == 1
                PlotDailyMisfires(fileLocations{x,1},engine, Mrun_time,...
                         MTotalTorque, MTotalspeed,...
                         total_Misfire_1, total_Misfire_2,...
                         total_Misfire_3, total_Misfire_4,...
                         total_Misfire_5, total_Misfire_6,1);
                PlotDailyMisfires(fileLocations{x,1},engine, Mrun_time,...
                         MTotalTorque, MTotalspeed,...
                         totalMisfire_1, totalMisfire_2,...
                         totalMisfire_3, totalMisfire_4,...
                         totalMisfire_5, totalMisfire_6,2);

            end
            catch
```

```
                set(handles.figure1, 'pointer', 'arrow');
                warning('Error Occured in plotting Misfires for entire time
interval');
                set(handles.PLOTpushbutton,'string', {'Error'});
            end
        end

    end

    if savegif == 1
        cd(gpath);
        try
        GIFfile = [fileLocations{x,1} ' Daily Plots.gif'];
        catch
            GIFfile = 'New GIF.gif';
        end
        fig = findobj('type','figure');
        for count=1:length(fig)-1
            frame = getframe(count); %this is figures open
            im = frame2im(frame);
            [A,map] = rgb2ind(im,256);
            if count == 1;
                imwrite(A,map,GIFfile,'gif','LoopCount',Inf,'DelayTime',2);
            else

imwrite(A,map,GIFfile,'gif','WriteMode','append','DelayTime',2);
            end
        end
    end

end %end of loop for all; went through each unit

if save == 1
    uistack(Tool)
    savepptallTL([path file]);
end

set(handles.PLOTpushbutton, 'BackgroundColor', [0, .8, .4]);
set(handles.PLOTpushbutton,'string', {'Plot'});
set(handles.figure1, 'pointer', 'arrow');
toc



% --- Executes on button press in dailyMisfirecheckbox.
function dailyMisfirecheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to dailyMisfirecheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in Misfirecheckbox.
function Misfirecheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to Misfirecheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of Misfirecheckbox


% --- Executes on button press in scatterSVDcheckbox.
function scatterSVDcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to scatterSVDcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of scatterSVDcheckbox


% --- Executes on button press in scatterTorqSpeedcheckbox.
function scatterTorqSpeedcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to scatterTorqSpeedcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of scatterTorqSpeedcheckbox


% --- Executes on button press in scatternormalizedcheckbox.
function scatternormalizedcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to scatternormalizedcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of
scatternormalizedcheckbox


% --- Executes on button press in normalizedRangescheckbox.
function normalizedRangescheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to normalizedRangescheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of normalizedRangescheckbox


% --- Executes on button press in normalvsDailytimecheckbox.
function normalvsDailytimecheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to normalvsDailytimecheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of
normalvsDailytimecheckbox


% --- Executes on button press in save2pptcheckbox.
function save2pptcheckbox_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to save2pptcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of save2pptcheckbox


% --- Executes on button press in extrapolateSVDcheckbox.
function extrapolateSVDcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to extrapolateSVDcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of extrapolateSVDcheckbox



function extrapSVDtoXYhours_Callback(hObject, eventdata, handles)
% hObject    handle to extrapSVDtoXYhours (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of extrapSVDtoXYhours as text
%        str2double(get(hObject,'String')) returns contents of
extrapSVDtoXYhours as a double


% --- Executes during object creation, after setting all properties.
function extrapSVDtoXYhours_CreateFcn(hObject, eventdata, handles)
% hObject    handle to extrapSVDtoXYhours (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in extraNormcheckbox.
function extraNormcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to extraNormcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of extraNormcheckbox



function editNormhours_Callback(hObject, eventdata, handles)
% hObject    handle to editNormhours (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editNormhours as text
%        str2double(get(hObject,'String')) returns contents of editNormhours
as a double


% --- Executes during object creation, after setting all properties.
function editNormhours_CreateFcn(hObject, eventdata, ~)
% hObject    handle to editNormhours (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in maxEnvelopecheckbox.
function maxEnvelopecheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to maxEnvelopecheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of maxEnvelopecheckbox


% --- Executes on button press in instantSVDcheckbox.
function instantSVDcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to instantSVDcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of instantSVDcheckbox


% --- Executes on button press in avgSVDcylinderscheckbox.
function avgSVDcylinderscheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to avgSVDcylinderscheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of avgSVDcylinderscheckbox


% --- Executes on button press in normalSVDtotalcheckbox.
function normalSVDtotalcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to normalSVDtotalcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of normalSVDtotalcheckbox


% --- Executes on button press in DutyPlotcheckbox.
function DutyPlotcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to DutyPlotcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DutyPlotcheckbox


% --- Executes on button press in SVD_milescheckbox.
function SVD_milescheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to SVD_milescheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of SVD_milescheckbox


% --- Executes on button press in dutyCombinedcheckbox.
function dutyCombinedcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to dutyCombinedcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of dutyCombinedcheckbox


% --- Executes on button press in MassAirFcheckbox.
function MassAirFcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to MassAirFcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of MassAirFcheckbox


% --- Executes on button press in extractmilescheckbox.
function extractmilescheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to extractmilescheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of extractmilescheckbox



function Extractmilesedit_Callback(hObject, eventdata, handles)
% hObject    handle to Extractmilesedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Extractmilesedit as text
%        str2double(get(hObject,'String')) returns contents of
Extractmilesedit as a double


% --- Executes during object creation, after setting all properties.
function Extractmilesedit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Extractmilesedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in warninglampcheckbox.
function warninglampcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to warninglampcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of warninglampcheckbox


% --- Executes on button press in savegifcheckbox.
function savegifcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to savegifcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of savegifcheckbox


% --- Executes on button press in DailyMAFcheckbox.
function DailyMAFcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to DailyMAFcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DailyMAFcheckbox


% --- Executes on button press in DailyMeshMAFcheckbox.
function DailyMeshMAFcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to DailyMeshMAFcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of DailyMeshMAFcheckbox


% --- Executes on button press in dailymeshcheckbox.
function dailymeshcheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to dailymeshcheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of dailymeshcheckbox


% --- Executes on button press in extractNORMmilescheckbox.
function extractNORMmilescheckbox_Callback(hObject, eventdata, handles)
% hObject    handle to extractNORMmilescheckbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of extractNORMmilescheckbox



function xmilesNormedit_Callback(hObject, eventdata, handles)
% hObject    handle to xmilesNormedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmilesNormedit as text
%        str2double(get(hObject,'String')) returns contents of xmilesNormedit
as a double


% --- Executes during object creation, after setting all properties.
function xmilesNormedit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmilesNormedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```