Computer Applications in Mechanical Engineering Education-Case 1: Diesel Engine Thermodynamics and Engine Performance Analysis with MATLAB

Inyeni A. Showers[1], Tonye K. Jack[2], O.M.O. Etebu[3]

[1, 2, 3]*Department of Mechanical Engineering, University of Port Harcourt*
Port Harcourt, 50004, Rivers State, Nigeria

***Abstract*—The article described, is a MATLAB developed Diesel Engine Thermodynamic and Engine test Performance analysis program, SDZEL. The program provides a fast calculation of diesel engine parameters and takes care of the tedium of repetitive hand calculations when parameter(s) values vary. The program capability is shown through a numerical example. The MATLAB codes are also provided to aid interested developers.**

***Keywords*— Diesel Engines, Compression Ignition Engines, Engine Thermodynamics Calculations, Engine Performance Analysis, Engineering Programs.**

I. Introduction

Concerns for exposing undergraduate mechanical engineering students to computing in terms of implementation of hands-on software applications to solution of real world engineering problems has dominated engineering curriculum developers' thoughts. Questions often centre around, what level of problem complexity at the undergraduate level should be set such that the student is able to fully acquire enough programming skills, lateral thinking ability, sound engineering judgement, discipline of focus in terms of achieving workable end results, self-starter status in terms of preparation for the work place of tomorrow, and enterprising spirit. Here is an experience of an Engineering Department's efforts. In a two part article, two such software programming projects are described. In this first paper, Case 1: A MATLAB program for a Diesel Engine thermodynamics and engine performance test analysis is described together with all equations used to develop the program. The MATLAB codes used to develop the program are also shown. The second article, Case 2, is a VISUAL BASIC developed helical gear sizing and stress load analysis program [1].

II. Equations Governing Diesel Engine Performance Parameters Evaluation

The general equations for the diesel engine performance test analysis are defined as follows:

A. Indicated Power

This is the power available in the cylinder, and its importance is in estimating the effects of atmospheric pressure and temperature [2]. It is defined by eq. (1):

$$IP = \frac{IMEP * SV * No.of\ cycles\ per\ minute}{60} \quad (1)$$

B. Brake Power

This is the usable drive power delivered by the engine crankshaft, and the actual work output at the load (driven) [3], [4]. It is defined by eq. (2):

$$BP = (\eta_{mech}).IP \quad (2)$$

C. Friction Power

Power Losses due to friction such as, at the working surfaces in bearings, piston rings and valves is the friction power [3]. It is the total power less the brake power as shown by eq. (3)

$$FP = IP - BP \quad (3)$$

D. Specific Fuel Consumption(SFC)

The rate of fuel consumption, per unit time, per power used [3]. It is of two forms, indicated specific fuel consumption (isfc), and brake specific fuel consumption (bsfc), depending, on if it is calculated with indicated power or brake power respectively. The unit reflects which power type is used – kg/ikWh, if indicated power is used, or kg/bkWh if brake power is used. Specific fuel consumption is useful in analysis of fuel economy [2]. Equation (4) shows the SFC, power relation.

$$SFC = \frac{Mass\ flow\ rate\ of\ fuel}{Power} \quad (4)$$

E. Engine Torque

Important from a driveshaft viewpoint [2], torque is the crankshaft turning effort to overcome friction and relates linearly with the brake power according to eq. (5)

$$T = \frac{60 * BP}{2\pi N} \quad (5)$$

F. Thermal Efficiency.

Relating to the heat content of the fuel used for work, the thermal efficiency of an engine is of two forms, the indicated thermal efficiency (ITE), which is the ratio of the total power, or total work done per cycle in the cylinders to the heat content available from the diesel fuel, and the brake thermal efficiency (BTE), which is the fraction of the heat content of the fuel converted into engine shaft work [2], [3].

1) Indicated Thermal Efficiency:

$$ITE = \frac{60 * IP}{m_f * CV} \tag{6}$$

2) Brake Thermal Efficiency:

$$BTE = \frac{60 * BP}{m_f * CV} \tag{7}$$

G. Mechanical Efficiency

Significant from the point of mechanical losses in an engine, the mechanical efficiency defined mathematically by eq. (8), is the ratio of the actual power delivered by the engine, to the total available in the engine cylinders [3], [5].

$$\eta_{mech} = \frac{BP}{IP} = \frac{IP - FP}{IP} \tag{8}$$

H. Mean Effective Pressure

This indicates an imagined value of the pressure acting on the piston face which if maintained throughout the power stroke cycle would give an equivalent of the amount of work to be done through the engine cycle of the engine in consideration [2], [3]. Mean effective pressure is used to compare engines of different sizes, since it is directly related to the torque per piston displacement [2], [4]. The two forms are, indicated mean effective pressure (IMEP), and the brake mean effective pressure (BMEP).

$$MEP = \frac{Work - Done - per - Cycle}{Swept - Volume} \tag{9}$$

Rajput [4], notes that since power of an engine is dependent on its size and speed, comparing engines on the basis of torque and power as defined by eq. (9) is not convenient. Thus, a more convenient form for the mean effective pressure is:

$$MEP = \frac{p_1 r^{\gamma}\left[\gamma(\rho - 1) - r^{1-\gamma}\left(\rho^{\gamma} - 1\right)\right]}{(\gamma - 1)(r - 1)} \tag{9a}$$

I. Diesel Efficiency

The theoretical diesel efficiency (air standard efficiency) is the ratio of the actual work done to the heat supplied, and defined by the eq. (10) as:

$$\eta_{diesel} = 1 - \frac{1}{\gamma(r)^{\gamma-1}}\left[\frac{\rho^{\gamma} - 1}{\rho - 1}\right] \tag{10}$$

Due to the high compression ratio of diesel engines, the efficiency of the diesel engine is practically higher than comparable petrol or spark-ignition engines [3].

J. Relative Efficiency or Efficiency Ratio

$$\eta_{relative} = \frac{\eta_{thermal}}{\eta_{air-standard}} \tag{11}$$

K. Specific Output

An indication of the brake output per piston displacement [4], the specific output is defined as,

$$Specific - Output = \frac{BP}{AL} \equiv cons\tan t \times BMEP \times N \tag{12}$$

The specific output is also referred to as power output per litre = (kW/litre) [6]. Thus, by eq. (12), given same piston displacement, and BMEP, an engine running at higher speed will give more power output [4].

III. THERMODYMIC PARAMETERS

A. Pressure, Temperature and Volume(PVT) at Salient Points of the Diesel Cycle

The standard Diesel Process Cycle [7] is shown in fig. (1).

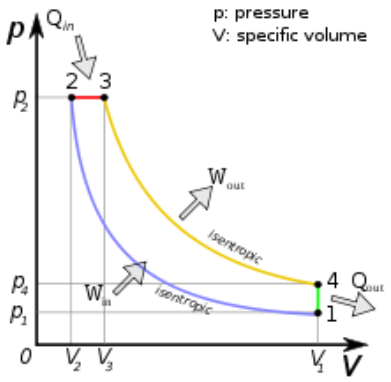

Fig. 1 A typical pressure-volume (PV) diagram of the Diesel Cycle.

Source: wikipedia [7]

The equations required for determining the unknown pressures, temperatures, and volumes at the cycle salient points of the diesel processes are available in the open literature and these are:

1) Process 1-2 – Adiabatic Compression:

Due to the work of compression of the piston compressing the working fluid [7], the following relation obtains:

$$P_1V_1^{\gamma} = P_2V_2^{\gamma} \quad (13)$$

Allowing for, P_2 to be obtained from eq. (13a):

$$\frac{P_2}{P_1} = \frac{V_1^{\gamma}}{V_2^{\gamma}} = (r)^{\gamma} \quad (13a)$$

And,

$$\gamma = \frac{C_p}{C_v} \quad (13b)$$

Thus, the compression ratio, $r = V_1 / V_2$ (13c)

Also,

$$\frac{T_2}{T_1} = \left(\frac{V_1}{V_2}\right)^{\gamma-1} = (r)^{\gamma-1} \quad (13d)$$

Giving, T_2.

The input volume, V_1, can be deduced from the Diesel cycle diagram to be:

$$V_1 = stroke\ volume + clearance\ volume$$

Or

$$V_1 = SV + CL = SV + V_2 \quad (13e)$$

Where,

$$stroke\ volume(SV) = \frac{\pi}{4}D^2\ L$$

$$V_2 = Clearance\ Volume = \frac{Stroke\ Volume}{r-1} \quad (13f)$$

(13g)

2) Process 2-3 – Heat Addition at Constant Pressure:

$$Q = m_2C_p(T_3 - T_2) \quad (14)$$

$$\frac{T_3}{T_2} = \frac{V_3}{V_2} = \rho \quad (14a)$$

$$T_3 = \rho.T_1.(r)^{\gamma-1} \quad (14b)$$

The cut-off ratio, ρ, relates to the compression ratio, r, by the relation of eq. (14c),

$$\rho = \frac{V_3}{V_2} = c(r-1)+1 \quad (14c)$$

$$V_3 = \rho.V_2 \quad (14d)$$

The percentage of stroke volume at cut off, c, is defined by eq. (14e):

$$percentage - cut - off\ - ratio, c = \frac{V_3 - V_2}{V_1 - V_2} = \frac{V_3 - V_2}{SV} = \frac{\rho - 1}{r - 1} \quad (14e)$$

Equation (14f) can be used to estimate the mass of the air, m_2, involved in the combustion process:

$$m_2 = \frac{P_2V_2}{RT_2} \quad (14f)$$

Fuel used in the process of combustion is then obtained by the relation of eq. (14g),

$$m_f = \frac{m_2}{AFR} \quad (14g)$$

Equations (14f) and (14g) are normalised for rate of flow per unit time by multiplying by factor, *KNn*.

Where, *AFR*, is the air-fuel ratio.

The pressure at cut-off, P_3, is equal to the pressure, P_2, at the beginning (-2-) of the combustion process, i.e.

$$P_3 = P_2 \quad (14h)$$

3) Process 3-4 – Adiabatic Expansion:

This process produces the usable torque, with the working fluid expanding on to the piston [7].

$$\frac{T_3}{T_4} = \left(\frac{V_4}{V_3}\right)^{\gamma-1} = \left(\frac{r}{\rho}\right)^{\gamma-1} \quad (15)$$

$$P_3V_3^{\gamma} = P_4V_4^{\gamma} \quad (15a)$$

$$\frac{P_4}{P_3} = \frac{V_3^{\gamma}}{V_4^{\gamma}} = \left[\frac{1}{(r/\rho)}\right]^{\gamma} \quad (15b)$$

4) Process 4-1 – Exhaust, Heat Rejection at constant volume:

$$T_4 = T_1.\rho^{\gamma} \quad (16)$$

$$V_4 = V_1 \quad (16b)$$

Fig. 2 Program SDZEL user interface for diesel cycle calculations

IV. Numerical Example

Analyse the performance requirements of a four-stroke, four-cylinder, Diesel Engine for the following parameters: L=300 mm; D=200 mm; P_1= 1 bar; T_1=27 ^oC; Percent cut-off = 8 percent; Compression ratio = 15; N = 760 rpm; Mechanical efficiency = 90 percent; Calorific value of the Diesel fuel = 34500 KJ/kg; specific heat ratio, γ =1.4; Gas constant, R = 287 KJ/kg/K; Air-fuel Ratio, AFR = 14.8 (this value more closely equates the AFR for Spark-ignition engines as against the range for the diesel compression-ignition engines of 18-25[4]. (See Conclusion).
(Some of the diesel parameter data were taken from [4], Example 3.21, pp. 115-118, as part of the program validation checks)

Fig. 3 Numerical example solution using SDZEL program

V. Conclusions

The program SDZEL is able to also conduct diesel engine parameter calculations for the two stroke diesel cycle. The program was tested extensively and found to be very active.

It is to be noted that the value of AFR as given in the numerical example, very closely equates that for spark-ignition engines. This strikes a balance for achieving high torque when the concept of excess air-factor, λ is taken into consideration [6]. In graphical plots provided by [6] for spark ignition engine management on the influence of air-factor and ignition timing on torque and specific fuel consumption, for excess air-factor, $\lambda>1$ which equates to that for diesel fuels, specifically, within the range $1<\lambda<1.2$, less fuel is consumed at lower torque, which equates to less power It would be interesting to look further at these thoughts, since as noted by [6], [4], the air/fuel ratio exercises a decisive effect on an engine's operating characteristics which depends on the torque and power as functions of speed [8]. In line with that, future work will require adding the option of conducting diesel fuel (s) combustion calculations, and extending the work to include dual combustion cycle.

Nomenclature

AFR = Air-fuel ratio
BMEP = Brake mean effective pressure, bar
BP = Brake power, kW
BSFC = Brake specific fuel consumption, Kg/BkWh
BTE = Brake thermal efficiency
c = Percentage of stroke volume at cut-off
CL = Clearance Volume, m^3
C_p = Specific heat at constant pressure, kJ/kg.K
C_v = Specific heat at constant volume, kJ/kg.K
CV = Fuel calorific value, kJ/Kg
FP = Friction power, kW
IP = Indicated power, kW
IMEP = Indicated mean effective pressure, bar
ISFC = Indicated specific fuel consumption, Kg/IkWh
ITE = Indicated thermal efficiency
K = a constant = 0.5 for four-stroke; =1 for two-stroke
m_a = Theoretical mass Flowrate of air, Kg/min=m_2.*(KNn)*
m_f = Mass Flowrate of fuel used, Kg/min= m_f.*(KNn)*
n=number of cylinders
N = Engine speed, rpm
R = Universal gas constant, J/Kg.K
SV = Stroke Volume, m^3

Greek letters:
ρ = cut-off ratio
r = Compression ratio
γ = Specific heat constant
$\eta_{air\text{-}standard}$ = Air-standard efficiency
η_{ith} *or RE* = Relative efficiency
η_{mech} *or ME* = Mechanical efficiency

VI. APPENDIX: MATLAB SOURCE CODES

```
function varargout= SDZEL_Mech_Dept_Uniport(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@SDZEL_Mech_Dept_Uniport_OpeningFcn, ...
                   'gui_OutputFcn',
@SDZEL_Mech_Dept_Uniport_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
    function
SDZEL_Mech_Dept_Uniport_OpeningFcn(hObject,
eventdata, handles, varargin)
    handles.output = hObject;
    % Update handles structure
    guidata(hObject, handles);

    function varargout =
SDZEL_Mech_Dept_Uniport_OutputFcn(hObject, eventdata,
handles)
    % Get default command line output from handles structure
    varargout{1} = handles.output;

    function D_Callback(hObject, eventdata, handles)
    D = str2double(get(hObject, 'String'));
    if isnan(D)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.D=D;
    guidata(hObject,handles)

    function D_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function L_Callback(hObject, eventdata, handles)
    L = str2double(get(hObject, 'String'));
    if isnan(L)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.L=L;
    guidata(hObject,handles)

    function L_CreateFcn(hObject, eventdata, handles)

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

    function T1_Callback(hObject, eventdata, handles)
    T1= str2double(get(hObject, 'String'));
    if isnan(T1)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.T1=T1;
    guidata(hObject,handles)
    function T1_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function P1_Callback(hObject, eventdata, handles)
    P1 = str2double(get(hObject, 'String'));
    if isnan(P1)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.P1=P1;
    guidata(hObject,handles)

    function P1_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function r_Callback(hObject, eventdata, handles)
    r = str2double(get(hObject, 'String'));
    if isnan(r)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.r=r;
    guidata(hObject,handles)
    function r_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function G_Callback(hObject, eventdata, handles)
    G = str2double(get(hObject, 'String'));
    if isnan(G)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.G=G;
    guidata(hObject,handles)
    function G_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function ME_Callback(hObject, eventdata, handles)
    ME = str2double(get(hObject, 'String'));
    if isnan(ME)
        set(hObject, 'String', '');
```

```
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.ME=ME;
    guidata(hObject,handles)
    function ME_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function CV_Callback(hObject, eventdata, handles)
    CV = str2double(get(hObject, 'String'));
    if isnan(CV)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.CV=CV;
    guidata(hObject,handles)
    function CV_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function AFR_Callback(hObject, eventdata, handles)
    AFR = str2double(get(hObject, 'String'));
    if isnan(AFR)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.AFR=AFR;
    guidata(hObject,handles)
    function AFR_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function N_Callback(hObject, eventdata, handles)
    N = str2double(get(hObject, 'String'));
    if isnan(N)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.N=N;
    guidata(hObject,handles)
    function N_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


    function IP_Callback(hObject, eventdata, handles)
    function IP_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function BP_Callback(hObject, eventdata, handles)
    function BP_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function C_Callback(hObject, eventdata, handles)
    C = str2double(get(hObject, 'String'));
    if isnan(C)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.C=C;
    guidata(hObject,handles)
    function C_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function K_Callback(hObject, eventdata, handles)
    function K_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function Calculate_Callback(hObject, eventdata, handles)
     K = str2double(get(handles.K, 'String'));
        handles.metricdata.K=K;
    SV = (pi/4)*handles.metricdata.D^2 * handles.metricdata.L;
    handles.metricdata.SV = SV ;
    SVr= round(SV/0.000001)*0.000001;
    set(handles.SV, 'String', SVr);
    CL = handles.metricdata.SV / (handles.metricdata.r - 1);
    handles.metricdata.CL = CL;
    CLr= round(CL/0.000001)*0.000001;
    set(handles.CL, 'String', CLr);
    V1 = handles.metricdata.SV + handles.metricdata.CL;
    handles.metricdata.V1 = V1;
    V1r= round(V1/0.00001)*0.00001;
    set(handles.V1, 'String', V1r);
    V2 = handles.metricdata.CL;
    handles.metricdata.V2 = V2;
    V2r= round(V2/0.00001)*0.00001;
    set(handles.V2, 'String', V2r);
    P11 = handles.metricdata.P1;
    handles.metricdata.P11 = P11;
    set(handles.P11, 'String', P11);
    P2 = handles.metricdata.P1 * ( handles.metricdata.r ^
handles.metricdata.G );
    handles.metricdata.P2 = P2;
    P2r= round(P2/0.01)*0.01;
```

```
    set(handles.P2, 'String', P2r);
    T11 = handles.metricdata.T1;
    handles.metricdata.T11 = T11;
    set(handles.T11, 'String', T11);
    P3 = handles.metricdata.P2;
    handles.metricdata.P3= P3;
    P3r= round(P3/0.01)*0.01;
    set(handles.P3, 'String', P3r);
    CR = (handles.metricdata.C * (handles.metricdata.r -1)) + 1;
    handles.metricdata.CR= CR;
    CRr= round(CR/0.000001)*0.000001;
    set(handles.CR, 'String', CRr);
    V3 = handles.metricdata.CR * handles.metricdata.V2;
    handles.metricdata.V3 = V3;
    V3r= round(V3/0.00001)*0.00001;
    set(handles.V3, 'String', V3r);
    T2 = handles.metricdata.T1 * (handles.metricdata.r
^(handles.metricdata.G -1));
    handles.metricdata.T2 = T2;
    T2r= round(T2/0.01)*0.01;
    set(handles.T2, 'String', T2r);
    T3 = handles.metricdata.T2 * (handles.metricdata.V3 /
handles.metricdata.V2);
    handles.metricdata.T3 = T3;
    T3r= round(T3/0.01)*0.01;
    set(handles.T3, 'String', T3r);
    V4 = handles.metricdata.V1;
    handles.metricdata.V4 = V4;
    V4r= round(V4/0.00001)*0.00001;
    set(handles.V4, 'String', V4r);
    P4 = handles.metricdata.P3 * ((handles.metricdata.V3 /
handles.metricdata.V4)^handles.metricdata.G);
    handles.metricdata.P4 = P4;
    P4r= round(P4/0.01)*0.01;
    set(handles.P4, 'String', P4r);
    T4 = handles.metricdata.T3 * ((handles.metricdata.V3 /
handles.metricdata.V4)^(handles.metricdata.G - 1));
    handles.metricdata.T4 = T4;
    T4r= round(T4/0.01)*0.01;
    set(handles.T4, 'String', T4r);
    imep = handles.metricdata.P1 *(handles.metricdata.r...
    ^ handles.metricdata.G) *(( handles.metricdata.G * ...
    ( handles.metricdata.CR - 1) - (handles.metricdata.r ^...
    (1- handles.metricdata.G)) * (( handles.metricdata.CR ^ ...
    handles.metricdata.G)-1))) / (( handles.metricdata.r - 1) *
( handles.metricdata.G - 1 ));
    handles.metricdata.imep = imep;
    imepr= round(imep/0.01)*0.01;
    set(handles.imep, 'String', imepr);
    IP=handles.metricdata.n *(handles.metricdata.imep *
handles.metricdata.SV * ...
    100*handles.metricdata.N *handles.metricdata.K)/60;
    handles.metricdata.IP= IP;
    IPr= round(IP/0.01)*0.01;
    set(handles.IP, 'String', IPr);
    BP=handles.metricdata.ME*handles.metricdata.IP;
    handles.metricdata.BP= BP;
    BPr= round(BP/0.01)*0.01;
    set(handles.BP, 'String', BPr);
    FP=handles.metricdata.IP-handles.metricdata.BP;
    handles.metricdata.FP= FP;
    FPr= round(FP/0.01)*0.01;
    set(handles.FP, 'String', FPr);
    bmep = handles.metricdata.ME*handles.metricdata.imep;
    bmepr= round(bmep/0.01)*0.01;
    set(handles.bmep, 'String', bmepr);
    handles.metricdata.bmep=bmep;
    Ma=(((handles.metricdata.K*handles.metricdata.N*handles
.metricdata.n*...
    handles.metricdata.P2*handles.metricdata.CL*10^5)/(handl
es.metricdata.R*...
    handles.metricdata.T2)));
    Mar= round(Ma/0.01)*0.01;
    set(handles.Ma, 'String', Mar);
    handles.metricdata.Ma=Ma;
    Mf=(handles.metricdata.Ma/(handles.metricdata.AFR));
    Mfr= round(Mf/0.01)*0.01;
       set(handles.Mf, 'String', Mfr);
       handles.metricdata.Mf=Mf;
    isfc=(handles.metricdata.Mf*60)/(handles.metricdata.IP);
    isfcr= round(isfc/0.01)*0.01;
       set(handles.isfc, 'String', isfcr);
       handles.metricdata.isfc=isfc;
    bsfc=(handles.metricdata.Mf*60)/(handles.metricdata.BP);
       bsfcr= round(bsfc/0.01)*0.01;
       set(handles.bsfc, 'String', bsfcr);
       handles.metricdata.bsfc=bsfc;
    ITE=60*IP/(handles.metricdata.Mf*handles.metricdata.CV
);
       ITEr= round(ITE/0.01)*0.01;
       set(handles.ITE, 'String', ITEr);
       handles.metricdata.ITE=ITE;
    BTE=60*BP/(handles.metricdata.Mf*handles.metricdata.C
V);
       BTEr= round(BTE/0.01)*0.01;
       set(handles.BTE, 'String', BTEr);
       handles.metricdata.BTE=BTE;
    DE=1 - (1 / (handles.metricdata.G * (handles.metricdata.r ^
(handles.metricdata.G - 1)))) ...
     * (((handles.metricdata.CR ^ handles.metricdata.G) - 1)/
( handles.metricdata.CR - 1 ));
       DEr= round(DE/0.01)*0.01;
       set(handles.DE, 'String', DEr);
       handles.metricdata.DE=DE;
    RE=handles.metricdata.BTE/handles.metricdata.DE;
       REr= round(RE/0.01)*0.01;
       set(handles.RE, 'String', REr);
    Tor=(handles.metricdata.BP*60*1000)/(2*pi*handles.metri
cdata.N);
       Torr= round(Tor/0.01)*0.01;
       set(handles.Tor, 'String', Torr);

    function Clear_Callback(hObject, eventdata, handles)
    set ( findobj( 0, 'style','edit'),'string','');
```

```
set(handles.Stroke, 'SelectedObject', handles.r1);
set(handles.K, 'String',1);

function Exit_Callback(hObject, eventdata, handles)
display Goodbye
close(gcbf)

function isfc_Callback(hObject, eventdata, handles)
function isfc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bsfc_Callback(hObject, eventdata, handles)
function bsfc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function imep_Callback(hObject, eventdata, handles)
function imep_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bmep_Callback(hObject, eventdata, handles)
function bmep_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ITE_Callback(hObject, eventdata, handles)
function ITE_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function BTE_Callback(hObject, eventdata, handles)
function BTE_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function DE_Callback(hObject, eventdata, handles)
function DE_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function RE_Callback(hObject, eventdata, handles)
function RE_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function FP_Callback(hObject, eventdata, handles)
function FP_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function T2_Callback(hObject, eventdata, handles)
function T2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function T3_Callback(hObject, eventdata, handles)
function T3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function T4_Callback(hObject, eventdata, handles)
function T4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function P2_Callback(hObject, eventdata, handles)
function P2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function P3_Callback(hObject, eventdata, handles)
function P3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function P4_Callback(hObject, eventdata, handles)
function P4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function V1_Callback(hObject, eventdata, handles)
function V1_CreateFcn(hObject, eventdata, handles)
```

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

    function V2_Callback(hObject, eventdata, handles)
    function V2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function V3_Callback(hObject, eventdata, handles)
    function V3_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function V4_Callback(hObject, eventdata, handles)
    function V4_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function n_Callback(hObject, eventdata, handles)
    n = str2double(get(hObject, 'String'));
    if isnan(n)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.n=n;
    guidata(hObject,handles)
    function n_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function Stroke_SelectionChangeFcn(hObject, eventdata,
handles)
    if (hObject == handles.r1)
        set(handles.K, 'String', '1');
    else
        set(handles.K, 'String', '0.5');
    end

    function R_Callback(hObject, eventdata, handles)
    R = str2double(get(hObject, 'String'));
    if isnan(R)
        set(hObject, 'String', '');
        errordlg('Input must be a number','Error');
    end
    handles.metricdata.R=R;
    guidata(hObject,handles)
    function R_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function SV_Callback(hObject, eventdata, handles)
    SV.setMaximumFractionDigits (2);
    function SV_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function CL_Callback(hObject, eventdata, handles)
    CL.setMaximumFractionDigits (2);
    function CL_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function CR_Callback(hObject, eventdata, handles)
    CR.setMaximumFractionDigits (2);
    function CR_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function T11_Callback(hObject, eventdata, handles)
    function T11_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function P11_Callback(hObject, eventdata, handles)
    function P11_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function Ma_Callback(hObject, eventdata, handles)
    function Ma_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    function Mf_Callback(hObject, eventdata, handles)
    function Mf_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

```
function Tor_Callback(hObject, eventdata, handles)
function Tor_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

ACKNOWLEDGMENT

This work is based on the final year engineering project work undertaken by Inyeni Amakuro Showers. The Department of Mechanical Engineering places great emphasis on introducing engineering students to the art of technical writing and producing project results of publishable quality. A mentoring approach is adopted by faculty members, thereby encouraging students to work alongside their project supervisors such that the expected high standards are achieved. The Department is appreciative of the effort made by Inyeni for spending the time in the demanding programming work, asking questions when in difficulty, and taking the initiative when required.

REFERENCES

[1] E. A. Chuku, T. K. Jack, H. U. Nwosu, "Computer applications in mechanical engineering education-case 2: helical gear design and analysis with visual basic," *International Journal of Emerging Trends and Technology (IJETT)*, 2014, (Accepted for publication)

[2] K. A. Carlson, "Internal combustion engines-1", *Machine Design*, pp. 156-170, Dec. 1954.

[3] N. K. Giri, *Automobile Technology*, Delhi, India:, Khanna, 2008

[4] R. K. Rajput, *Internal Combustion Engines*, 2nd ed., Delhi, India: Laxmi, 2007

[5] S. Singh, *Mechanical Engineer's Handbook*, Delhi, India: Khanna, 2008

[6] H. Bauer, Ed., *Automotive Handbook*, 4th ed. , Stuttgart, Germany: Bosch, 1996

[7] (2014) The Wikipedia website. [Online]. Available: http://en.wikipedia.org/wiki/Diesel_cycle

[8] T. D. Gillespie, *Fundamentals of vehicle dynamics*, SAE, 1992