computation

MDPI

Article

MATLAB Application for Determination of 12 Combustion Products, Adiabatic Temperature and Laminar Burning Velocity: Development, Coding and Explanation

Roberto Franco Cisneros * and Freddy Jesus Rojas *

Faculty of Mechanical Engineering, Pontificia Universidad Católica del Perú, Lima 15088, Peru
* Correspondence: a20150535@pucp.pe (R.F.C.); fjrojas@pucp.edu.pe (F.J.R.)

Abstract: The determination of the characteristics and main combustion properties of fuels is necessary for post-implementation in different applications. Among the most important combustion properties of a fuel are the combustion products, flame temperature and laminar burning velocity. Therefore, this paper describes the step-by-step development and coding of a MATLAB application that can determine 12 combustion products, flame temperature and laminar burning velocity in order to understand the logic of calculus procedure, so any user would be able to make improvements of new functionalities (add more fuels, add more combustion products, etc.). The numerical procedure and methods (Gaussian elimination, Taylor Series and Newton–Raphson) parallel with their implementation as code lines for the development of the application are carried out using flow charts. In addition, simulations in Ansys Chemkin were performed and included in the application as part of the results comparison. It was found that: (1) The MATLAB Application codification and development were successfully explained in detail, (2) the functions and execution sequence are described by using flow charts and code extract, (3) the application is available to everyone for modifications, (4) the application can only be used for hydrocarbons fuels, (5) the application execution time registered was less than 8 s.

Keywords: numerical methods; combustion products; flame temperature; burning velocity; Gaussian elimination method; Taylor series; Newton–Raphson method

Citation: Cisneros, R.F.; Rojas, F.J. MATLAB Application for Determination of 12 Combustion Products, Adiabatic Temperature and Laminar Burning Velocity: Development, Coding and Explanation. *Computation* **2024**, *12*, 189. https://doi.org/10.3390/computation12090189

Academic Editor: Sivasankaran Sivanandam

Received: 1 August 2024
Revised: 6 September 2024
Accepted: 9 September 2024
Published: 16 September 2024

1. Introduction

The simulation of combustion processes remains a key component in the development and design of new engines, power sources, and machinery. Among the different studies that facilitate the acquisition of new results, there are numerical procedures, experimental tests, and software simulations. Software tools such as Ansys Chemkin, Cantera, and other applications are necessary for obtaining an initial understanding of the combustion process as they provide critical data on new systems under development.

Within the previous research conducted by various authors, different programs and codes have been used for many applications. In 1989, Zhu and Egolfopoulos made a numerical simulation of one-dimensional flame propagation by using a flame code developed by Kee and coworkers that has in its program multicomponent diffusion relations, mass conservation equation and a C1 kinetic mechanism developed by the same author [1]. Starting in the 2000s, Tsotsis and Egolfopoulos used the PREMIX code developed in FORTRAN for determining the mole fractions of the products, and the laminar burning velocities of different methane–CO_2 mixtures [2]. In 2012, Erjiang et al. studied the effect of composition on laminar burning velocities of $H_2/CO/N_2/CO_2$/air mixtures using experimental and numerical studies. In the latest one, they simulated using PREMIX code with CHEMKIN II a freely propagating adiabatic premixed planar flame [3]. In this study, the Li Mechanism, Davis Mechanism, USCII Mechanism, GRI-MECH 3.0 Mechanism and Sun Mechanism kinetics were used [3]. Seven years later, Longkai et al. did a numerical analysis of the effect

of CO_2 on combustion characteristics of laminar premixed methane/air flames. Similar to previous works, they used the PREMIX code for the calculation of the one-dimensional laminar premixed combustion characteristics of (CH_4 + CO_2)/air mixtures with different CO_2 contents and equivalent ratios [4]. Later, in 2020, Xie et al. employed the Premixed Laminar Flame-speed Calculation model to calculate the one-dimensional free propagation flame with the PREMIX code of Chemkin package for a numerical analysis on the effects of CO_2 dilution on the laminar burning velocity of premixed methane/air flame with elevated initial temperature and pressure [5]. All equations of PREMIX code use can be found in the Chemkin-Pro 19.0 theory manual [6].

Programs and programming languages such as MATLAB, PYTHON or FORTRAN are preferred by the scientific community for a wide range of applications as they have enough features for the analysis process. In the MATLAB App Designer, it is possible to create various apps to observe and analyze the results of different studies in a more intuitive manner. For this study, the main contribution is the detailed description and explanation of the development of the MATLAB Application. Considering the main reference to the FORTRAN code developed by Olikara and Borman [7], which already includes the numerical methodology for calculating flame temperature and combustion products, a new package of codes was created in MATLAB. This new one compared with the Olikara and Borman FORTRAN has the novelty of including the determination of the laminar burning velocity. Finally, a MATLAB Application was developed including the new package of codes and all the features detailed in the present work. Code lines and flow charts are presented to help readers understand the programming logic for future improvements of new tools, new fuels or new combustion products. This work is a sequence of the published article "Determination of 12 Combustion Products, Flame Temperature and Laminar Burning Velocity of Saudi LPG Using Numerical Methods Coded in a MATLAB Application" that previously has presented the validation of the app [8]. The methodology of the project is explained, and the results obtained are compared with different experimental and numerical articles for some fuels (methane, propane and natural gas).

2. Materials and Methods

The numerical method was originally developed in FORTRAN in 1975 by Olikara and Borman in their paper titled "A Computer Program for Calculating Properties of Equilibrium Combustion Products with Some Applications to I. C. Engines" [7]; and modified to include the determination of the laminar burning velocity by using MATLAB R2021b for the codes package and the MATLAB App Designer for the development of the application. The simulation results included in the application were carried out in Ansys Chemkin 2022 R2. Figure 1 presents the methodology diagram for the PI075 Project.

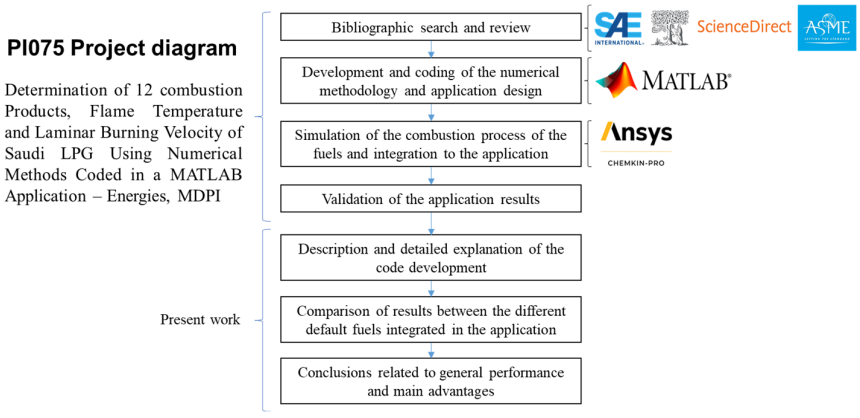


Figure 1. Methodology for the present work.

A computer with an AMD Ryzen 7 5700G with Radeon Graphics 3.80 GHz processor with 32 GB RAM and Windows 11 Enterprise was used for this purpose.

3. Development of the MATLAB Application

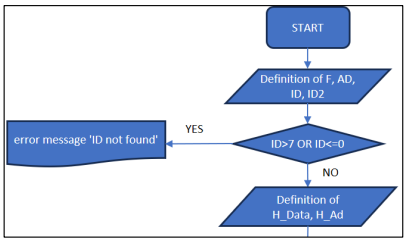
The application designed in MATLAB is divided into three main parts; for each one, a flowchart of how the code works was constructed which can be found in Appendix A. The first one (Figure A1) receives all the necessary inputs to start the iteration process until the adiabatic temperature is found. Secondly, in Figure A2, all the numerical matrices and derivative-necessary operations are performed, so the combustion products and other important values are obtained. Finally, Figure A3 explains how the initialization of the process of the application works. Theorical energy equations and statements have already been explained and presented in the previously published article referenced as [6].

3.1. Part One: Develop of the "New_code" File

This first part (Code S1) starts with the necessary inputs for the iteration process: the equivalence ratio (F), the fraction of the diluent (AD, if applicable), the ID of the fuel (ID assigned in File S1 called "REACTANTS ENTHALPY" with the properties of the fuel) and the ID of the diluent (ID2 assigned in File S2 called "DILUENT ENTHALPY" with the properties of the diluent if applies). The first step is a verification of the existence of the introduced fuel ID in the database File S1 as shown in Figure 2. Then, it is possible to define the variables H_Data (Enthalpies values) and H_Ad (Enthalpies diluents values), as these variables will store all the data related to the reactant and the diluent (if applicable), respectively.

```
%NEW CODE FOR DETERMINATION OF 12 COMBUSTION PRODUCTS, FLAME TEMPERATURE
%AND LAMINAR BURNING VELOCITY
function [T,V,X]=New_code(F,AD,ID,ID2)
if ID>7 || ID<=0
    error 'ID not found'
end
```

(**a**)



(**b**)

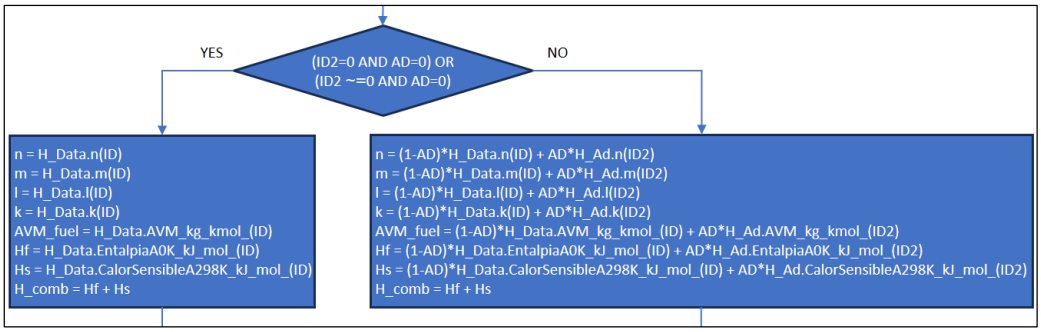
Figure 2. Initial verification of fuel ID and lecture of properties in New_code.m: (**a**) Code line, (**b**) Flowchart.

After the verification, it is possible to calculate the fuel enthalpy with the ratio of equivalence, the specified fuel and the fraction of diluent (if applicable) by using the variables H_Data and H_Ad that were defined previously, as shown in Figure 3.

```
if (ID2==0 && AD==0) || (ID2~=0 && AD==0)
n = H_Data.n(ID);
m = H_Data.m(ID) ;
l = H_Data.l(ID) ;
k = H_Data.k(ID) ;
AVM_fuel = H_Data.AVM_kg_kmol_(ID) ;
Hf = H_Data.EntalpiaA0K_kJ_mol_(ID) ;
Hs = H_Data.CalorSensibleA298K_kJ_mol_(ID) ;
H_comb= Hf + Hs;

else
n = (1-AD)*H_Data.n(ID) + AD*H_Ad.n(ID2);
m = (1-AD)*H_Data.m(ID) + AD*H_Ad.m(ID2);
l = (1-AD)*H_Data.l(ID) + AD*H_Ad.l(ID2);
k = (1-AD)*H_Data.k(ID) + AD*H_Ad.k(ID2);

AVM_fuel = (1-AD)*H_Data.AVM_kg_kmol_(ID) + AD*H_Ad.AVM_kg_kmol_(ID2);
Hf = (1-AD)*H_Data.EntalpiaA0K_kJ_mol_(ID)...
+ AD*H_Ad.EntalpiaA0K_kJ_mol_(ID2);
Hs = (1-AD)*H_Data.CalorSensibleA298K_kJ_mol_(ID)...
+ AD*H_Ad.CalorSensibleA298K_kJ_mol_(ID2);
H_comb= Hf + Hs;
end
```

(**a**)

Figure 3. *Cont.*



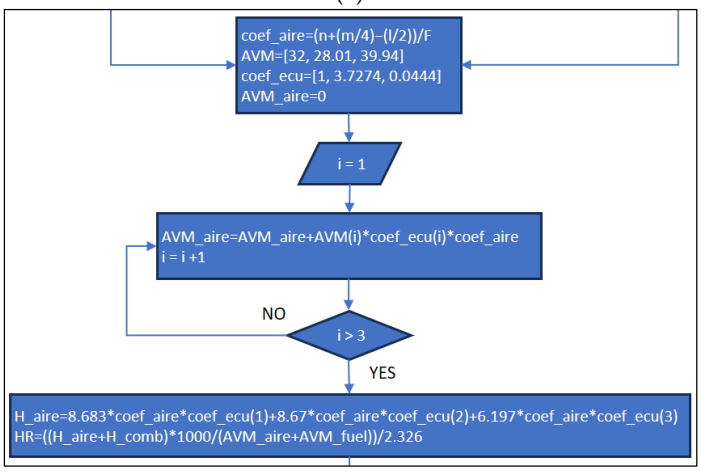
(**b**)

Figure 3. Determination of the fuel enthalpy in New_code.m: (**a**) Code line, (**b**) Flowchart.

The next step is to calculate the air enthalpy using the ratio of equivalence, the coefficients related to the fuel (n, m, l, k) and the AVM which is defined as a matrix. The whole reactant enthalpy (air plus fuel) is calculated as shown in Figure 4.

```
36  %Air
37  coef_aire=(n+(m/4)-(l/2))/F;
38  %    O2    N2     AR
39  AVM=[32, 28.01, 39.94];
40  coef_ecu=[1, 3.7274, 0.0444];
41  AVM_aire=0;
42  for i=1:3
43  AVM_aire=AVM_aire+AVM(i)*coef_ecu(i)*coef_aire;
44
45  end
46  H_aire=8.683*coef_aire*coef_ecu(1)+8.67*coef_aire*coef_ecu(2)...
47  + 6.197*coef_aire*coef_ecu(3);
48
49  HR=((H_aire+H_comb)*1000/(AVM_aire+AVM_fuel))/2.326; %BTU/lbm
```

(**a**)



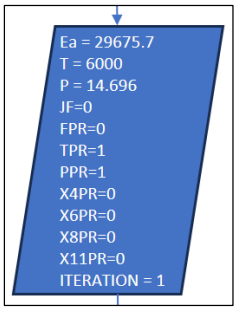
(**b**)

Figure 4. Determination of the air enthalpy and whole reactants enthalpy: (**a**) Code line, (**b**) Flowchart.

Following the previous step, it is necessary to set the Activation energy (Ea), which is defined as equal to 29,875.7 cal/mol, the first estimated temperature (T) 6000 R, the work pressure (P) 1 atm or 14.696 psi and the variables necessary for the iteration process required to determine the 12 combustion products, as shown in Figure 5.

```
% Ea: Energy activation cal/mol
Ea=29675.7;
% T: Temperature in Rankine
T=6000;
% P: Pressure in PSIA = 14.696
P=14.696;
%Variables for determination of the molar fractions and derivatives
JF=0; % JF=0: another iteration is done, JF=1: no more iterations are done
FPR=0; %previous ratio result - constant
TPR=1; %previous temperature result
PPR=1; %previous pressure result - constant
X4PR=0; %previous iteration X4 fraction result
X6PR=0; %previous iteration X6 fraction result
X8PR=0; %previous iteration X8 fraction result
X11PR=0; %previous iteration X11 fraction result
```

(**a**)



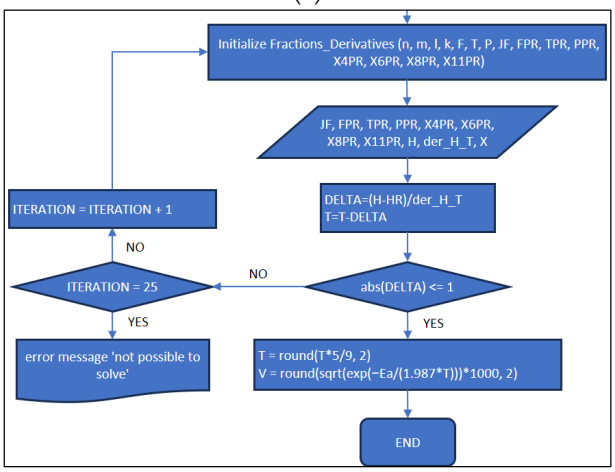
(**b**)

Figure 5. Definition of the activation energy, first estimate temperature, pressure and variables for the iteration process: (**a**) Code line, (**b**) Flowchart.

These variables and all the inputs that are defined in Figure 5 are introduced in the function Fractions_Derivatives where the iteration process begins. DELTA is calculated using the reactants (HR) and Products Enthalpy (H) difference, divided by the derivative of enthalpy respect temperature (der_H_T). The loop finish when DELTA is less equal to 1 so the flame temperature (T) and laminar burning velocity (V) can be calculated using the Mallard and Le Chateller method as shown in Figure 6.

```
for ITERATION=1:25

[JF,FPR,TPR,PPR,X4PR,X6PR,X8PR,X11PR,H,der_H_T,X]=...
Fractions_Derivatives(n,m,l,k,F,T,P,JF,FPR,TPR,PPR,X4PR,X6PR,X8PR,X11PR);

DELTA=(H-HR)/der_H_T;
T=T-DELTA;
if(abs(DELTA)<=1)
    %Determination of the laminar burning velocity using Mallard and Le
    %Chatelier method
    format shortG
    T=round(T*5/9,2);
    V=round(sqrt(exp(-Ea/(1.987*T)))*1000,2);
    break
end

end
end
```

(**a**)



(**b**)

Figure 6. Iteration process for the determination of the product fractions, flame temperature and laminar burning velocity: (**a**) Code line, (**b**) Flowchart.

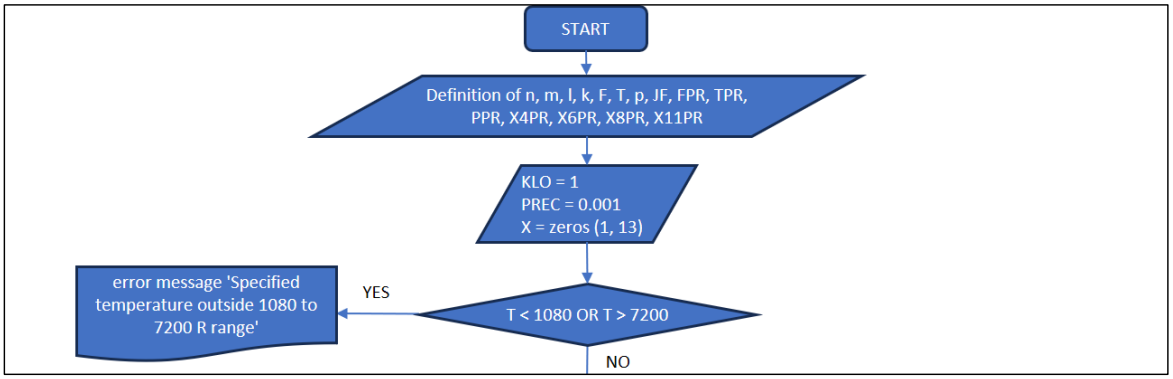
3.2. Part Two: Develop of the "Fractions_Derivatives" File

For part two (Code S2), the first action performed by the program is to define the size of the matrix for combustion products (X), the precision for the calculations and verify

if the first estimated temperature defined in the first part (6000 R) or the calculated for a specific iteration is between the range of 1080–7200 R as shown in Figure 7.

```
%Fractions_Derivatives
function [JF,FPR,TPR,PPR,X4PR,X6PR,X8PR,X11PR,H,der_H_T,X]=Fractions_Derivatives(n ...
    ,m,l,k,F,T,p,JF,FPR,TPR,PPR,X4PR,X6PR,X8PR,X11PR)
KLO=1;
syms OX
PREC=1*10^(-3);
X=zeros(1,13);
if (T<1080 || T>7200)
    error 'Specified temperature outside 1080 to 7200 R range';
end
```

(**a**)



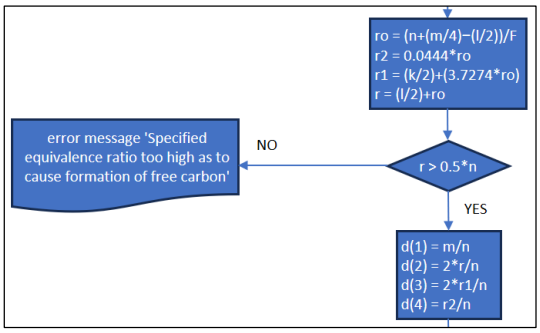
(**b**)

Figure 7. First definitions and verification of the initial defined estimated temperature: (**a**) Code line, (**b**) Flowchart.

The next step in the process is the definition and determination of the constants related to the combustion reaction balance. In addition, verification of the equivalent ratio is performed to ensure that it is not too high which can cause the formation of free carbon because this would not let the application solve the equation system, as shown in Figure 8.

```
%Determination of Constants
ro=(n+(m/4)-(l/2))/F;
r2=0.0444*ro;
r1=(k/2)+(3.7274*ro);
r=(l/2)+ro;
if (r>0.5*n)
    d(1)=m/n;
    d(2)=2*r/n;
    d(3)=2*r1/n;
    d(4)=r2/n;
else
 error ['Specified equivalence ratio too high as to cause formation of free carbon'] ;
end
```

(**a**)



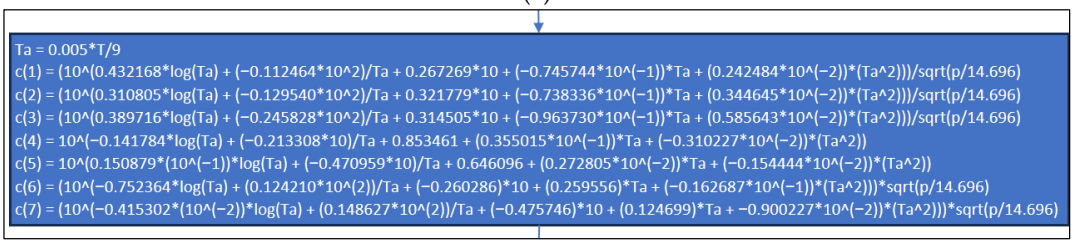
(**b**)

Figure 8. Definition and determination of constants and verification of the equivalent ratio: (**a**) Code line, (**b**) Flowchart.

A temperature Ta is defined as an input in the equilibrium constants at constant pressure (Kp) for each reaction which is shown in Figure 9.

```
%Kp
% Ta:Temperature used for ajusted the coefficients
Ta=0.005*T/9;
%Constant of Reaction 0.5(H2) --> H (C1)
c(1)=(10^(0.432168*log(Ta) + (-0.112464*10^2)/Ta + 0.267269*10 + ...
    (-0.745744*10^(-1))*Ta + (0.242484*10^(-2))*(Ta^2)))/sqrt(p/14.696);

%Constant of Reaction 0.5(O2) --> O (C2)
c(2)=(10^(0.310805*log(Ta) + (-0.129540*10^2)/Ta + 0.321779*10 + ...
    (-0.738336*10^(-1))*Ta + (0.344645*10^(-2))*(Ta^2)))/sqrt(p/14.696);

%Constant of Reaction 0.5(N2) --> N (C3)
c(3)=(10^(0.389716*log(Ta) + (-0.245828*10^2)/Ta + 0.314505*10 + ...
    (-0.963730*10^(-1))*Ta + (0.585643*10^(-2))*(Ta^2)))/sqrt(p/14.696);

%Constant of Reaction 0.5(O2)+0.5(H2) --> OH (C5)
c(4)=10^(-0.141784*log(Ta) + (-0.213308*10)/Ta + 0.853461 + ...
    (0.355015*10^(-1))*Ta + (-0.310227*10^(-2))*(Ta^2));

%Constant of Reaction 0.5(N2)+0.5(O2) --> NO (C7)
c(5)=10^(0.150879*(10^(-1))*log(Ta) + (-0.470959*10)/Ta + 0.646096 + ...
    (0.272805*10^(-2))*Ta + (-0.154444*10^(-2))*(Ta^2));

%Constant of Reaction H2+0.5(O2) --> H2O (C9)
c(6)=(10^(-0.752364*log(Ta) + (0.124210*10^(2))/Ta + (-0.260286)*10 + ...
    (0.259556)*Ta + (-0.162687*10^(-1))*(Ta^2)))*sqrt(p/14.696);

%Constant of Reaction CO+0.5(O2) --> CO2 (C10)
c(7)=(10^(-0.415302*(10^(-2))*log(Ta) + (0.148627*10^(2))/Ta + ...
    (-0.475746)*10 + (0.124699)*Ta + (-0.900227*10^(-2))*(Ta^2)))*sqrt(p/14.696);
```

(**a**)



(**b**)

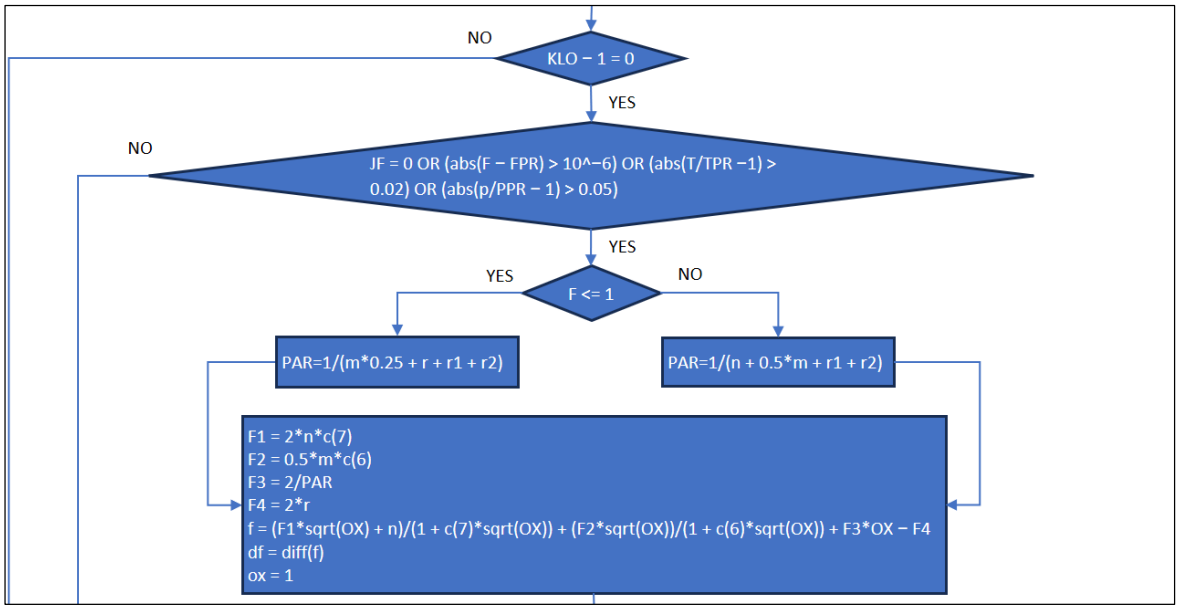
Figure 9. Definition of the equilibrium constants at constant pressure (Kp): (**a**) Code line, (**b**) Flowchart.

After the Kp constants are defined, depending on the ratio of equivalence that is going to be used, an "if" condition is stated for ratios less equal 1 and for greater than 1. With the "if" condition a first estimated value of 'PAR' (that represents x_{13}) is calculated with variables 'F1', 'F2', 'F3', 'F4' and 'f' (general linear function). Later, the first differential equation of 'f' is calculated ('df') and ox (first estimate value of x_8) is defined in order to start the Newton–Raphson iteration process as shown in Figure 10.

```
if KLO-1==0
    if JF==0 || (abs(F-FPR)>10^-6) || (abs(T/TPR -1)>0.02) || (abs(p/PPR -1)>0.05)
        if F<=1
          PAR=1/(m*0.25+r+r1+r2);
        else
          PAR=1/(n+0.5*m+r1+r2);
        end
        F1=2*n*c(7);
        F2=0.5*m*c(6);
        F3=2/PAR;
        F4=2*r;
        f=(F1*sqrt(OX)+n)/(1+c(7)*sqrt(OX)) + (F2*sqrt(OX))/(1+c(6)*sqrt(OX))...
        +F3*OX-F4;
%Newton Raphson used for the determination of the zero:
        df=diff(f);
        OX=1;
```

(**a**)

Figure 10. *Cont.*



(**b**)

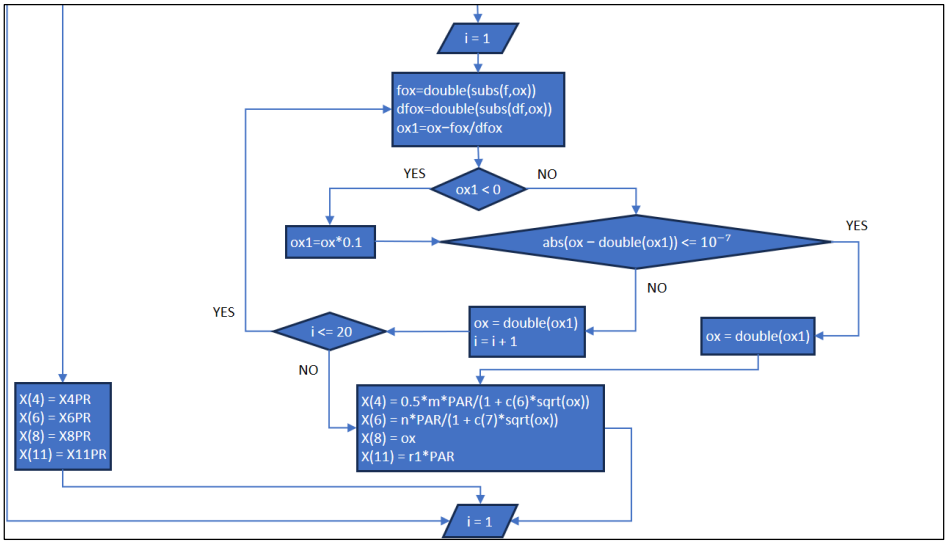
Figure 10. First estimation of 'PAR' (X13), the definition of 'df' and 'ox' variables for Newton–Raphson iteration process: (**a**) Code line, (**b**) Flowchart.

With 'df' it is possible to determine the first estimated values of x_4, x_6, x_8, x_{11}. The Newton–Raphson method is applied in order to find the zero of equation 'f'. Two variables 'fox' and 'dfox' are defined in order to store the value of 'ox' replaced in 'f' and in 'df', respectively. The Newton–Raphson process is set to take 20 iterations maximum as a limit as during simulations for all the pre-set fuels, and the iteration process mostly ends at the 8th iteration. This number of iterations also helps in the case of new fuels that might be added and could require more iteration time. After the iteration process successfully fulfils the tolerance (less equal 0.0000001), the values of x_4, x_6, x_8, and x_{11} are calculated using the previously defined equations as shown in Figure 11.

```
        for i=1:20
        fox=double(subs(f,ox));
        dfox=double(subs(df,ox));
        ox1=ox-fox/dfox;
        %In case the first obtained value is negative, ox1 is reduced to it's tenth
        if ox1<0
        ox1=ox*0.1;
        end
        if (abs(ox-double(ox1))<=0.0000001)
        ox=double(ox1);
        break
        else
        ox=double(ox1);
        end
        end
%X4, X6, X8 and X11 first estimated result obtained:
        X(4)=0.5*m*PAR/(1+c(6)*sqrt(ox));
        X(6)=n*PAR/(1+c(7)*sqrt(ox));
        X(8)=ox;
        X(11)=r1*PAR;
    else
        X(4)=X4PR;
        X(6)=X6PR;
        X(8)=X8PR;
        X(11)=X11PR;
    end
end
```

(**a**)

Figure 11. *Cont.*



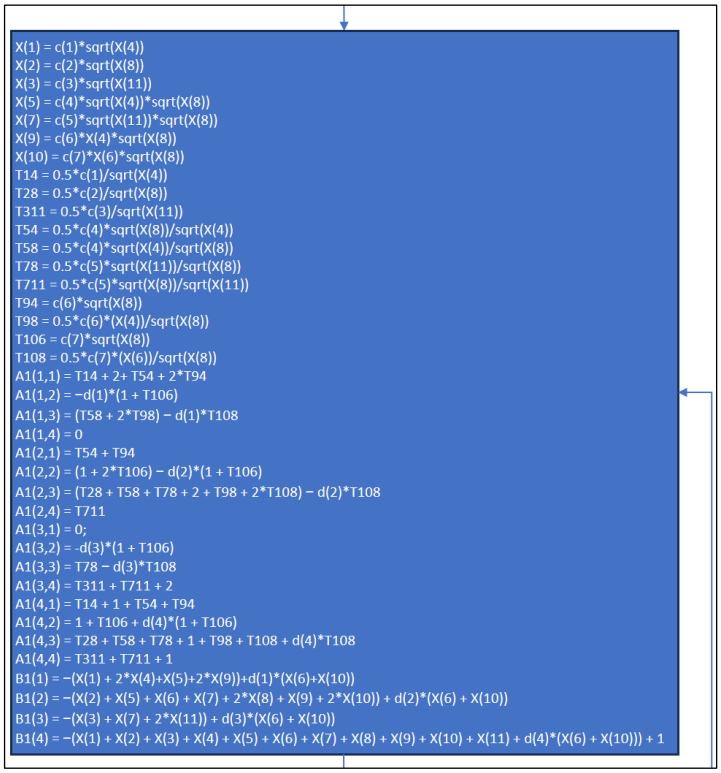
(**b**)

Figure 11. Newton–Raphson method: (**a**) Code line, (**b**) Flowchart.

After a first estimate of the values x_4, x_6, x_8, x_{11}, it is possible to define the other products using these values. In addition, the values of the derivatives for each molar product respect x_4, x_6, x_8, x_{11} and the four functions (B_j) are also defined as shown in Figure 12 in order to form a matrix equation system.

```
%Equations for other products are defined using the constants Kp:
X(1)=c(1)*sqrt(X(4));
X(2)=c(2)*sqrt(X(8));
X(3)=c(3)*sqrt(X(11));
X(5)=c(4)*sqrt(X(4))*sqrt(X(8));
X(7)=c(5)*sqrt(X(11))*sqrt(X(8));
X(9)=c(6)*X(4)*sqrt(X(8));
X(10)=c(7)*X(6)*sqrt(X(8));
%Functions and derivatives of every product by other are defined:
T14=0.5*c(1)/sqrt(X(4));
T28=0.5*c(2)/sqrt(X(8));
T311=0.5*c(3)/sqrt(X(11));
T54=0.5*c(4)*sqrt(X(8))/sqrt(X(4));
T58=0.5*c(4)*sqrt(X(4))/sqrt(X(8));
T78=0.5*c(5)*sqrt(X(11))/sqrt(X(8));
T711=0.5*c(5)*sqrt(X(8))/sqrt(X(11));
T94=c(6)*sqrt(X(8));
T98=0.5*c(6)*(X(4))/sqrt(X(8));
T106=c(7)*sqrt(X(8));
T108=0.5*c(7)*(X(6))/sqrt(X(8));
A1(1,1)=T14+2+T54+2*T94;
A1(1,2)=-d(1)*(1+T106);
A1(1,3)=(T58+2*T98)-d(1)*T108;
A1(1,4)=0;
A1(2,1)=T54+T94;
A1(2,2)=(1+2*T106)-d(2)*(1+T106);
A1(2,3)=(T28+T58+T78+2+T98+2*T108)-d(2)*T108;
A1(2,4)=T711;
A1(3,1)=0;
A1(3,2)=-d(3)*(1+T106);
A1(3,3)=T78-d(3)*T108;
A1(3,4)=T311+T711+2;
A1(4,1)=T14+1+T54+T94;
A1(4,2)=1+T106+d(4)*(1+T106);
A1(4,3)=T28+T58+T78+1+T98+T108+d(4)*T108;
A1(4,4)=T311+T711+1;
B1(1)=-(X(1)+2*X(4)+X(5)+2*X(9))+d(1)*(X(6)+X(10));
B1(2)=-(X(2)+X(5)+X(6)+X(7)+2*X(8)+X(9)+2*X(10))+d(2)*(X(6)+X(10));
B1(3)=-(X(3)+X(7)+2*X(11))+d(3)*(X(6)+X(10));
B1(4)=-(X(1)+X(2)+X(3)+X(4)+X(5)+X(6)+X(7)+X(8)+X(9)+X(10)+X(11)+ ...
    d(4)*(X(6)+X(10)))+1;
```

(**a**)



(**b**)

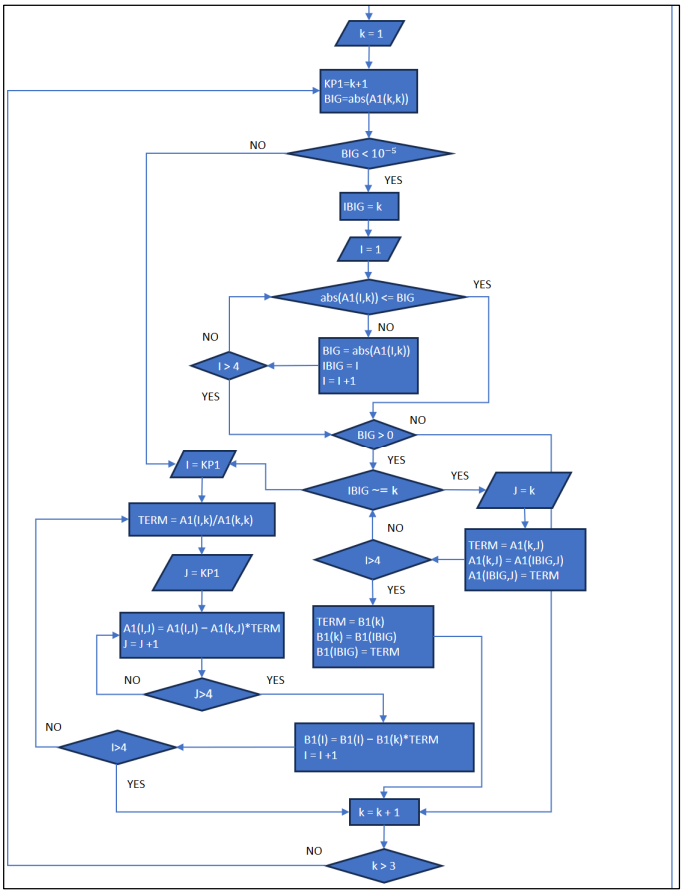
Figure 12. Definition of equations for the other combustion products and their derivatives for matrix equation system: (**a**) Code line, (**b**) Flowchart.

Once the matrix system is defined, the Gaussian elimination method is applied using row interchange only if the pivot point is smaller than 10^{-5}. This, in the code, is a for loop

and has some restrictions in order to obtain the modified matrix. All of this process is shown in Figure 13.

```
for k=1:3
    KP1=k+1;
    BIG=abs(A1(k,k));
    if (BIG<10^-5)
        IBIG=k;
        for I=KP1:4
            if abs(A1(I,k))<=BIG
                break
            else
                BIG=abs(A1(I,k));
                IBIG=I;
            end
        end
        if (BIG>0)
            if (IBIG~=k)
                for J=k:4
                    TERM=A1(k,J);
                    A1(k,J)=A1(IBIG,J);
                    A1(IBIG,J)=TERM;
                end
                TERM=B1(k);
                B1(k)=B1(IBIG);
                B1(IBIG)=TERM;
            else
                for I=KP1:4
                    TERM=A1(I,k)/A1(k,k);
                    for J=KP1:4
                        A1(I,J)=A1(I,J)-A1(k,J)*TERM;
                    end
                    B1(I)=B1(I)-B1(k)*TERM;
                end
            end
        end
    else
         for I=KP1:4
                    TERM=A1(I,k)/A1(k,k);
                    for J=KP1:4
                        A1(I,J)=A1(I,J)-A1(k,J)*TERM;
                    end
                    B1(I)=B1(I)-B1(k)*TERM;
         end
    end
end
```

(**a**)



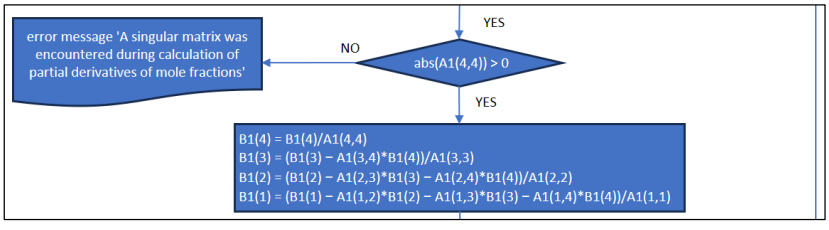
(**b**)

Figure 13. Resolution of the matrix equation system by Gaussian elimination method using row interchange: (**a**) Code line, (**b**) Flowchart.

In the case that the existence of a singular matrix occurs, an error message will be displayed as shown in Figure 14.

```
if (abs(A1(4,4))>0)
B1(4)=B1(4)/A1(4,4);
B1(3)=(B1(3)-A1(3,4)*B1(4))/A1(3,3);
B1(2)=(B1(2)-A1(2,3)*B1(3)-A1(2,4)*B1(4))/A1(2,2);
B1(1)=(B1(1)-A1(1,2)*B1(2)-A1(1,3)*B1(3)-A1(1,4)*B1(4))/A1(1,1);
else
  error ['A singular matrix was encountered during calculatin of partial derivatives of mole fractions'];
end
```

(**a**)



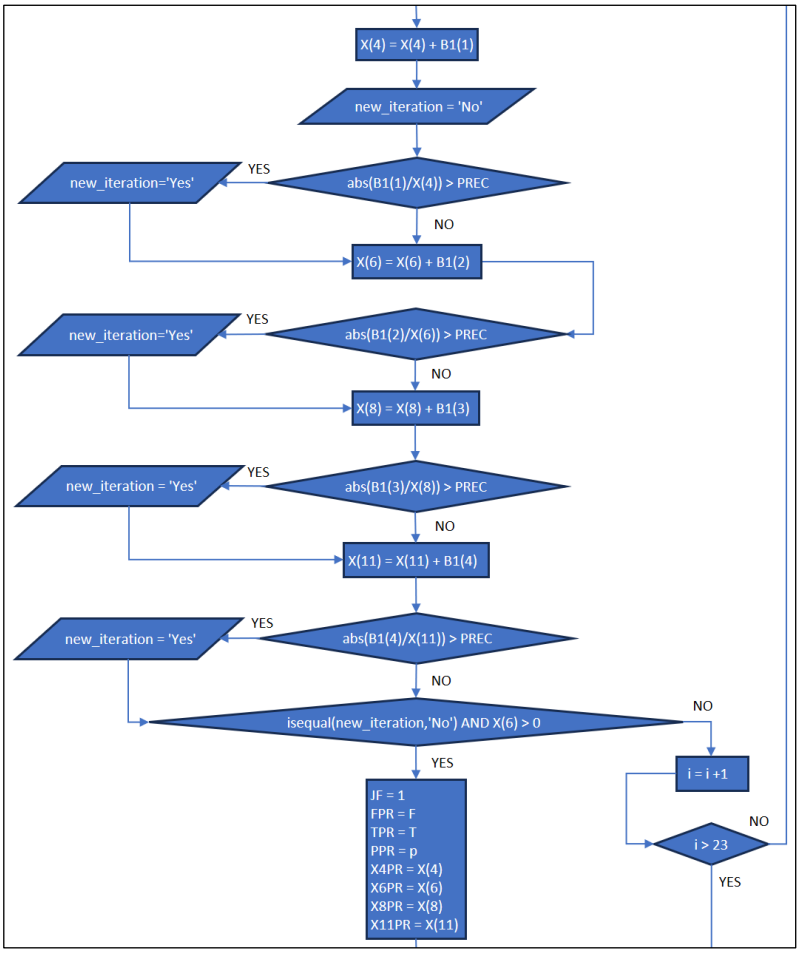
(**b**)

Figure 14. Final verification of the existence of a singular matrix in the matrix system: (**a**) Code line, (**b**) Flowchart.

After doing the verification, in case there is no error, the values of x_4, x_6, x_8, and x_{11} are calculated one more time in order to adjust them and the tolerance is verified. Moreover, the value of the variable "new_iteration" is "No" until the end, which will be the end of the iteration process in the matrix, and the other products are calculated as shown in Figure 15. If the opposite happens, another iteration is carried out.

```
%The values of X4, X6, X8 and X11 are recalculated
%in order to adjust them
X(4)=X(4)+B1(1);
new_iteration='No';
%Verification of the tolerance
if abs(B1(1)/X(4))>PREC
    new_iteration='Yes';
end
X(6)=X(6)+B1(2);
if abs(B1(2)/X(6))>PREC
    new_iteration='Yes';
end
X(8)=X(8)+B1(3);
if abs(B1(3)/X(8))>PREC
    new_iteration='Yes';
end
X(11)=X(11)+B1(4);
if abs(B1(4)/X(11))>PREC
    new_iteration='Yes';
end
%In case the tolerance is OK, the other products
%are calculated
if isequal(new_iteration,'No') && X(6)>0
JF=1;
FPR=F;
TPR=T;
PPR=p;
X4PR=X(4);
X6PR=X(6);
X8PR=X(8);
X11PR=X(11);
    break %end of for loop
end
end
```

(**a**)



(**b**)

Figure 15. Verification of the results tolerance and final calculation of the products: (**a**) Code line, (**b**) Flowchart.

After finding the molar products, it is necessary to recalculate the previous variables (derivatives of molar products and functions F1, F2, F3 and F4) in order to have them available for post-calculus as shown in Figure 16.

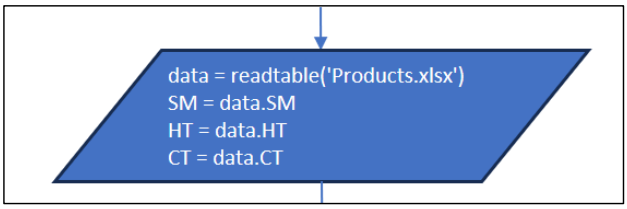
Moreover, the properties of products (molar mass—SM, enthalpy—HT, specific heat capacity—CT) are added into the program by reading File S4 which contains them, as shown in Figure 17. The properties will be used for the final determination of the specific enthalpy of the whole products.

```
%Recalculate the variables for post necessary calculus
T14=0.5*c(1)/sqrt(X(4));
T28=0.5*c(1)/sqrt(X(8));
T311=0.5*c(3)/sqrt(X(11));
T54=0.5*c(4)*sqrt(X(8))/sqrt(X(4));
T58=0.5*c(4)*sqrt(X(4))/sqrt(X(8));
T78=0.5*c(5)*sqrt(X(11))/sqrt(X(8));
T711=0.5*c(5)*sqrt(X(8))/sqrt(X(11));
T94=c(6)*sqrt(X(8));
T98=0.5*c(6)*(X(4))/sqrt(X(8));
T106=c(7)*sqrt(X(8));
T108=0.5*c(7)*(X(6))/sqrt(X(8));
A1(1,1)=T14+2+T54+2*T94;
A1(1,2)=-d(1)*(1+T106);
A1(1,3)=(T58+2*T98)-d(1)*T108;
A1(1,4)=0;
A1(2,1)=T54+T94;
A1(2,2)=(1+2*T106)-d(2)*(1+T106);
A1(2,3)=(T28+T58+T78+2+T98+2*T108)-d(2)*T108;
A1(2,4)=T711;
A1(3,1)=0;
A1(3,2)=-d(3)*(1+T106);
A1(3,3)=T78-d(3)*T108;
A1(3,4)=T311+T711+2;
A1(4,1)=T14+1+T54+T94;
A1(4,2)=1+T106+d(4)*(1+T106);
A1(4,3)=T28+T58+T78+1+T98+T108+d(4)*T108;
A1(4,4)=T311+T711+1;
```

(**a**) (**b**)

Figure 16. Recalculation of variables for post operations procedure: (**a**) Code line, (**b**) Flowchart.

```
%Properties of products are read
data = readtable('Products.xlsx');
SM = data.SM; %Molar Mass
HT = data.HT; %Enthalpy
CT = data.CT; %Specific Heat capacity
```



(**a**) (**b**)

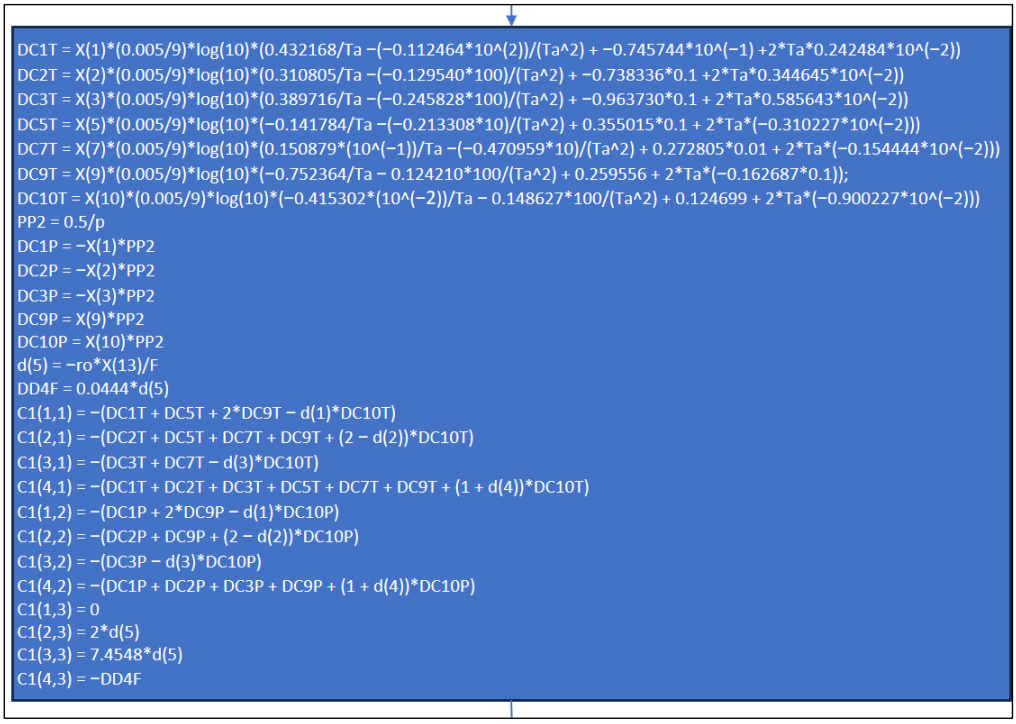
Figure 17. Introduction of properties and products data to the program: (**a**) Code line, (**b**) Flowchart.

The next step is to define the derivative respect temperature (DCT), pressure (DCP) and ratio (DD4F), as shown in Figure 18, in order to determine the derivative of enthalpy respect temperature (der_H_T), which depends on the value of the derivative of mole fraction respect temperature and the derivative of molar mass respect temperature.

After defining the derivative respect temperature, pressure and ratio, and setting the new equation systems (C1 equations matrix), the Gaussian elimination using maximum pivot strategy is used for the solving the matrix equation system as shown in Figure 19.

```
%Definition of derivatives respect temperature, pressure and ratio
%Respect Temperature
DC1T=X(1)*(0.005/9)*log(10)*(0.432168/Ta -(-0.112464*10^(2))/(Ta^2) + -0.745744*10^(-1) +2*Ta*0.242484*10^(-2));
DC2T=X(2)*(0.005/9)*log(10)*(0.310805/Ta -(-0.129540*100)/(Ta^2) + -0.738336*0.1 +2*Ta*0.344645*10^(-2));
DC3T=X(3)*(0.005/9)*log(10)*(0.389716/Ta -(-0.245828*100)/(Ta^2) + -0.963730*0.1 + 2*Ta*0.585643*10^(-2));
DC5T=X(5)*(0.005/9)*log(10)*(-0.141784/Ta -(-0.213308*10)/(Ta^2) + 0.355015*0.1 + 2*Ta*(-0.310227*10^(-2)));
DC7T=X(7)*(0.005/9)*log(10)*(0.150879*(10^(-1))/Ta -(-0.470959*10)/(Ta^2) + 0.272805*0.01 + 2*Ta*(-0.154444*10^(-2)));
DC9T=X(9)*(0.005/9)*log(10)*(-0.752364/Ta - 0.124210*100/(Ta^2) + 0.259556 + 2*Ta*(-0.162687*0.1));
DC10T=X(10)*(0.005/9)*log(10)*(-0.415302*(10^(-2))/Ta - 0.148627*100/(Ta^2) + 0.124699 + 2*Ta*(-0.900227*10^(-2)));
%Respect Pressure
PP2=0.5/p;
DC1P=-X(1)*PP2;
DC2P=-X(2)*PP2;
DC3P=-X(3)*PP2;
DC9P=X(9)*PP2;
DC10P=X(10)*PP2;
%Respect Ratio
d(5)=-ro*X(13)/F;
DD4F=0.0444*d(5);
C1(1,1)=-(DC1T+DC5T+2*DC9T-d(1)*DC10T);
C1(2,1)=-(DC2T+DC5T+DC7T+DC9T+(2-d(2))*DC10T);
C1(3,1)=-(DC3T+DC7T-d(3)*DC10T);
C1(4,1)=-(DC1T+DC2T+DC3T+DC5T+DC7T+DC9T+(1+d(4))*DC10T);
C1(1,2)=-(DC1P+2*DC9P-d(1)*DC10P);
C1(2,2)=-(DC2P+DC9P+(2-d(2))*DC10P);
C1(3,2)=-(DC3P-d(3)*DC10P);
C1(4,2)=-(DC1P+DC2P+DC3P+DC9P+(1+d(4))*DC10P);
C1(1,3)=0;
C1(2,3)=2*d(5);
C1(3,3)=7.4548*d(5);
C1(4,3)=-DD4F;
```

(**a**)

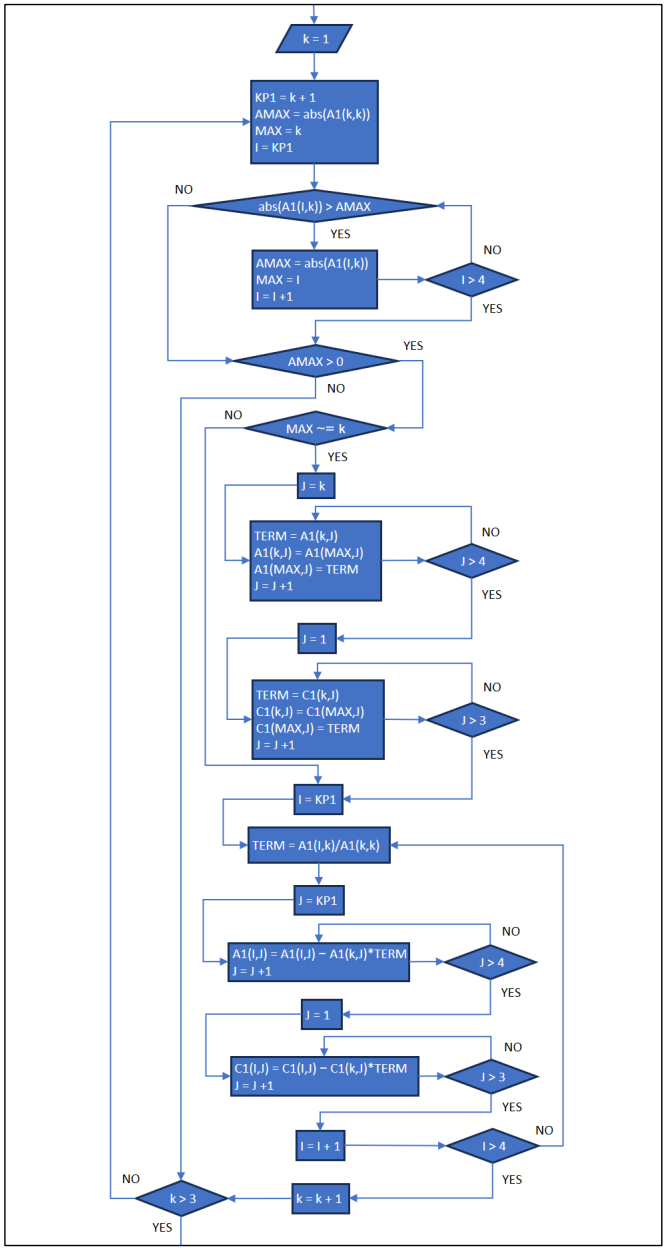


(**b**)

Figure 18. Definition of derivatives respect temperature, pressure and ratio: (**a**) Code line, (**b**) Flowchart.

```
for k=1:3
KP1=k+1;
AMAX=abs(A1(k,k));
MAX=k;
    for I=KP1:4
        if (abs(A1(I,k))>AMAX)
            AMAX=abs(A1(I,k));
            MAX=I;
        else
           break
        end
    end
    if AMAX>0
        if MAX~=k
            for J=k:4
                TERM=A1(k,J);
                A1(k,J)=A1(MAX,J);
                A1(MAX,J)=TERM;
            end
            for J=1:3
                TERM=C1(k,J);
                C1(k,J)=C1(MAX,J);
                C1(MAX,J)=TERM;
            end
        end

        for I=KP1:4
            TERM=A1(I,k)/A1(k,k);
            for J=KP1:4
                A1(I,J)=A1(I,J)-A1(k,J)*TERM;
            end
            for J=1:3
                C1(I,J)=C1(I,J)-C1(k,J)*TERM;
            end
        end
    end
end
```

(**a**)



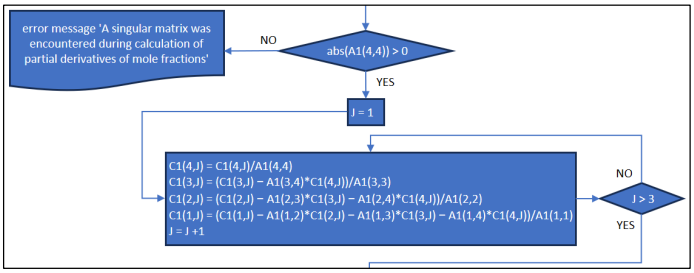
(**b**)

Figure 19. Resolution of the matrix equation system by Gaussian elimination method using maximum pivot strategy: (**a**) Code line, (**b**) Flowchart.

After completing the maximum numbers of iterations, a verification of the value of A1(4,4) is performed in order to validate if any singular matrix was found. In the case that it passes the validation, each element of the matrix C1 is calculated, because they contain the derivatives of each property. As the matrix has three columns and four rows, an iteration is necessary for each column in order to calculate each element as shown in Figure 20. In the opposite case, an error message is displayed.

```
334   if(abs(A1(4,4))>0)
335       for J=1:3
336           C1(4,J)=C1(4,J)/A1(4,4);
337           C1(3,J)=(C1(3,J)-A1(3,4)*C1(4,J))/A1(3,3);
338           C1(2,J)=(C1(2,J)-A1(2,3)*C1(3,J)-A1(2,4)*C1(4,J))/A1(2,2);
339           C1(1,J)=(C1(1,J)-A1(1,2)*C1(2,J)-A1(1,3)*C1(3,J)-A1(1,4)*C1(4,J))/A1(1,1);
340       end
341   else
342     error ['A singular matrix was encountered during calculation of partial derivatives of mole fractions'];
343   end
```

(**a**)



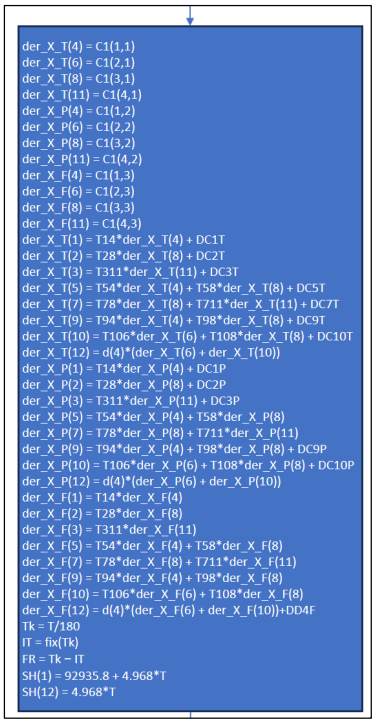
(**b**)

Figure 20. Verification of the existence of a singular matrix in the system: (**a**) Code line, (**b**) Flowchart.

Then, the partial derivatives with respect temperature, pressure and ratio are calculated using each element of the C1 solved matrix, and 'Tk', 'IT', 'FR', 'SH (1)', 'SH (12)' are calculated and defined for the determination of product enthalpy (H) and average molar mass (AVM) as shown in Figure 21.

```
344   der_X_T(4)=C1(1,1);
345   der_X_T(6)=C1(2,1);
346   der_X_T(8)=C1(3,1);
347   der_X_T(11)=C1(4,1);
348   der_X_P(4)=C1(1,2);
349   der_X_P(6)=C1(2,2);
350   der_X_P(8)=C1(3,2);
351   der_X_P(11)=C1(4,2);
352   der_X_F(4)=C1(1,3);
353   der_X_F(6)=C1(2,3);
354   der_X_F(8)=C1(3,3);
355   der_X_F(11)=C1(4,3);
356   der_X_T(1)=T14*der_X_T(4)+DC1T;
357   der_X_T(2)=T28*der_X_T(8)+DC2T;
358   der_X_T(3)=T311*der_X_T(11)+DC3T;
359   der_X_T(5)=T54*der_X_T(4)+T58*der_X_T(8)+DC5T;
360   der_X_T(7)=T78*der_X_T(8)+T711*der_X_T(11)+DC7T;
361   der_X_T(9)=T94*der_X_T(4)+T98*der_X_T(8)+DC9T;
362   der_X_T(10)=T106*der_X_T(6)+T108*der_X_T(8)+DC10T;
363   der_X_T(12)=d(4)*(der_X_T(6)+der_X_T(10));
364   der_X_P(1)=T14*der_X_P(4)+DC1P;
365   der_X_P(2)=T28*der_X_P(8)+DC2P;
366   der_X_P(3)=T311*der_X_P(11)+DC3P;
367   der_X_P(5)=T54*der_X_P(4)+T58*der_X_P(8);
368   der_X_P(7)=T78*der_X_P(8)+T711*der_X_P(11);
369   der_X_P(9)=T94*der_X_P(4)+T98*der_X_P(8)+DC9P;
370   der_X_P(10)=T106*der_X_P(6)+T108*der_X_P(8)+DC10P;
371   der_X_P(12)=d(4)*(der_X_P(6)+der_X_P(10));
372   der_X_F(1)=T14*der_X_F(4);
373   der_X_F(2)=T28*der_X_F(8);
374   der_X_F(3)=T311*der_X_F(11);
375   der_X_F(5)=T54*der_X_F(4)+T58*der_X_F(8);
376   der_X_F(7)=T78*der_X_F(8)+T711*der_X_F(11);
377   der_X_F(9)=T94*der_X_F(4)+T98*der_X_F(8);
378   der_X_F(10)=T106*der_X_F(6)+T108*der_X_F(8);
379   der_X_F(12)=d(4)*(der_X_F(6)+der_X_F(10))+DD4F;
380   %Determination of the AVM, Cp of gases and  enthalpy
381   Tk=T/180;
382   IT=fix(Tk);
383   FR=Tk-IT;
384   SH(1)=92935.8+4.968*T;
385   SH(12)=4.968*T;
```

(**a**)

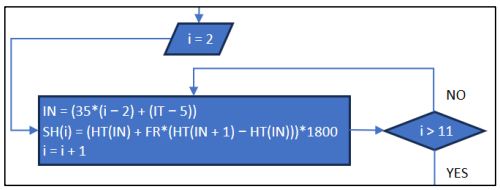


(**b**)

Figure 21. Recalculation of partial derivatives: (**a**) Code line, (**b**) Flowchart.

The enthalpy of each product is calculated and stored in a single row matrix (SH) as shown in Figure 22.

```
386    for i=2:11
387        IN=(35*(i-2)+(IT-5));
388        SH(i)=(HT(IN)+FR*(HT(IN+1)-HT(IN)))*1800;
389    end
```

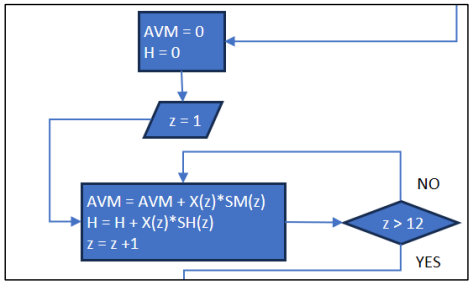


(**a**) (**b**)

Figure 22. Determination of the single row matrix SH: (**a**) Code line, (**b**) Flowchart.

By using the enthalpy of each product SH, the average molar mass (AVM) and product enthalpy (HR) are calculated as shown in Figure 23.

```
390    AVM=0;
391    H=0;
392    for z=1:12
393        AVM=AVM+X(z)*SM(z);
394        H=H+X(z)*SH(z);
395    end
```

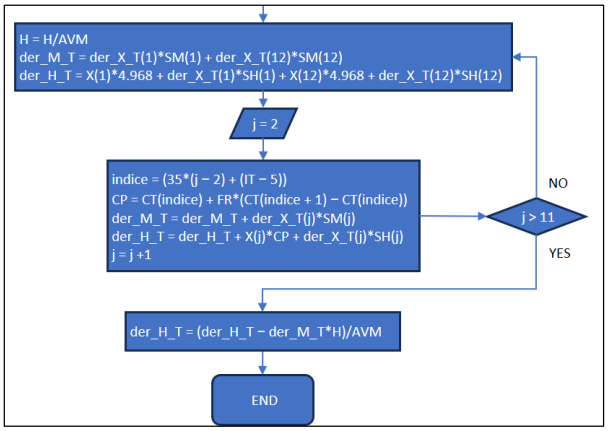


(**a**) (**b**)

Figure 23. Determination of average molar mass (AVM) and products enthalpy: (**a**) Code line, (**b**) Flowchart.

The specific product enthalpy is calculated dividing the product enthalpy by the average molar mass, and the partial derivatives with respect temperature are calculated for enthalpy and molar mass just for the first and last products (1 and 12). Iterating between 2 and 11 values, the CP (constant pressure heat) is calculated, and it is possible to determine the final value of the partial derivative of enthalpy with respect temperature that is needed for the DELTA determination in New_code.m as shown in Figure 24.

```
396    H=H/AVM;
397    %Determination of the partial derivatives respect T, P and F for H and M
398    der_M_T=der_X_T(1)*SM(1)+der_X_T(12)*SM(12);
399    der_H_T=X(1)*4.968+der_X_T(1)*SH(1)+X(12)*4.968+der_X_T(12)*SH(12);
400    for j=2:11
401        indice=(35*(j-2)+(IT-5));
402        CP=CT(indice)+FR*(CT(indice+1)-CT(indice));
403        der_M_T=der_M_T+der_X_T(j)*SM(j);
404        der_H_T=der_H_T+X(j)*CP+der_X_T(j)*SH(j);
405    end
406    der_H_T=(der_H_T-der_M_T*H)/AVM;
407    end
```



(**a**) (**b**)

Figure 24. Determination of CP and partial derivative of enthalpy respect temperature: (**a**) Code line, (**b**) Flowchart.

3.3. Part Three: Develop of the MATLAB Application File

In this part, the code and flowchart when the application is initialized is explained. In addition, the tools of the MATLAB Application are explained along with how they work through the programming code. When the application is started, the properties that

variables that can be used in any function defined in the application are specified as shown in Figure 25.

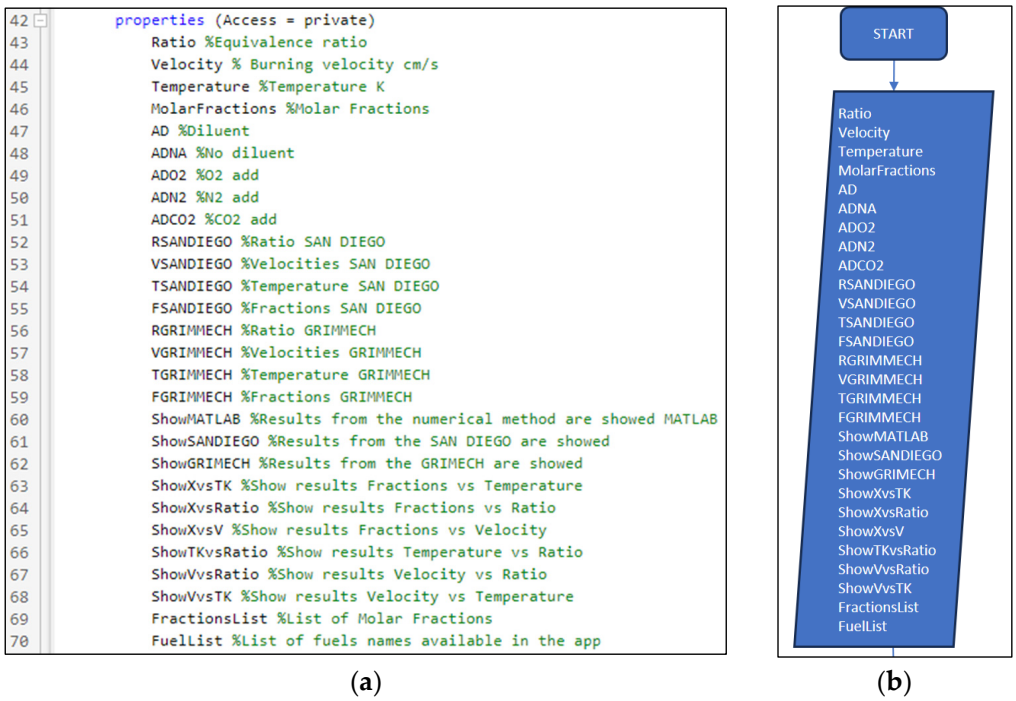


Figure 25. Properties definition in the application: (**a**) Code line, (**b**) Flowchart.

Then, when the application is started, 'Label2' is defined to show the current date and the molar fraction list item is set. The default value assigned to the variable corresponding to the diluent app.AD is 0 as shown in Figure 26.

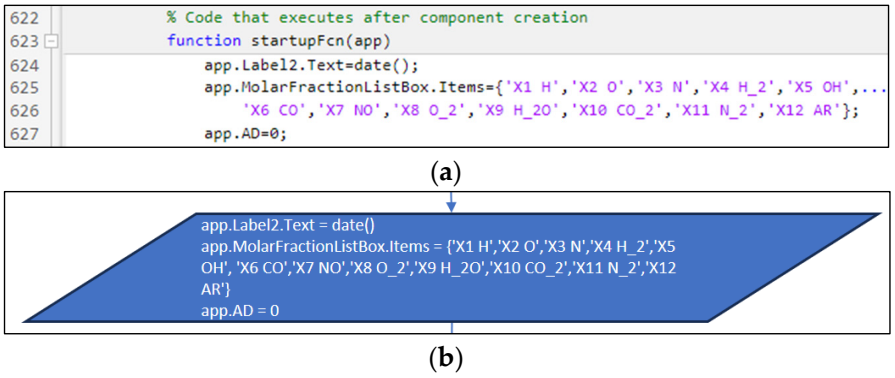
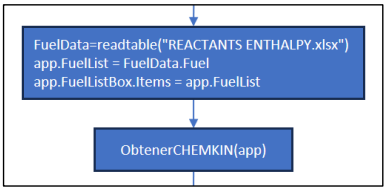


Figure 26. Definition of molar fraction list items: (**a**) Code line, (**b**) Flowchart.

The reactant enthalpy excel archive is also used during the determination of the products, and temperature and burning velocity are used for defining the list of fuels that will be considered in the application. In addition to this, the app function 'ObtenerCHEMKIN' is initialized in order to obtain the results from the simulation that are stored in an excel archive as shown in Figure 27.

```
628            FuelData=readtable("REACTANTS ENTHALPY.xlsx");
629            app.FuelList=FuelData.Fuel;
630            app.FuelListBox.Items=app.FuelList;
631            ObtenerCHEMKIN(app);
```

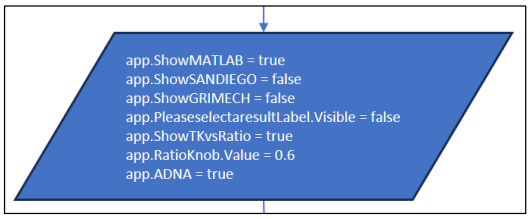


(**a**) (**b**)

Figure 27. Definition of the fuel list and lecture of the Ansys Chemkin simulations results: (**a**) Code line, (**b**) Flowchart.

Moreover, variables like app.ShowMATLAB, app.ShowSANDIEGO (presents Ansys Chemkin simulation results using San Diego mechanism storage in File S3 [9]), app.ShowGRIMECH (presents Ansys Chemkin simulation results using GRI-MECH 3.0 mechanism results storage in File S3 [10]) are defined with the default values as presented in Figure 28. Other variables like app.PleaseselectaresultLabel enable a message, requiring the user to select a plot, in the case that any option of the plot is selected. The app.RatioKnob value defines the numerical ratio that will be set up in the tools that will be presented in the following steps.

```
632            app.ShowMATLAB=true;
633            app.ShowSANDIEGO=false;
634            app.ShowGRIMECH=false;
635            app.PleaseselectaresultLabel.Visible=false;
636            app.ShowTKvsRatio=true;
637            app.RatioKnob.Value=0.6;
```

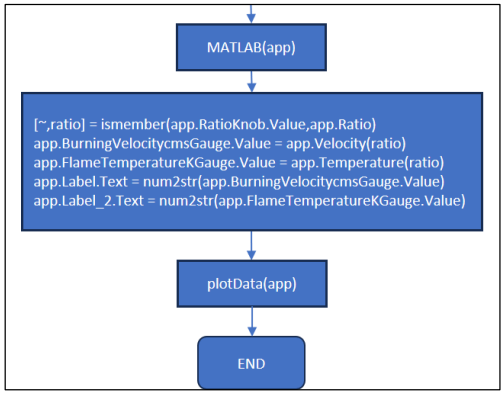


(**a**) (**b**)

Figure 28. Setting of the variables for plotting: (**a**) Code line, (**b**) Flowchart.

Finally, MATLAB function is called in order to obtain the results from the New_code.m and Fractions_Derivatives.m archives. In addition, the final gauges and tools labels are set with the result values and the function plotData is called for the purpose of creating the plot with the results obtained. This final part is shown in Figure 29.

```
639            MATLAB(app);
640            [~,ratio]=ismember(app.RatioKnob.Value,app.Ratio);
641            app.BurningVelocitycmsGauge.Value=app.Velocity(ratio);
642            app.FlameTemperatureKGauge.Value=app.Temperature(ratio);
643            app.Label.Text=num2str(app.BurningVelocitycmsGauge.Value);
644            app.Label_2.Text=num2str(app.FlameTemperatureKGauge.Value);
645            plotData(app);
646        end
```



(**a**) (**b**)

Figure 29. MATLAB and plot functions: (**a**) Code line, (**b**) Flowchart.

Finally, when the application is executed, the screen shown in Figure 30 will appear, and all the tools are identified and explained in Table 1.

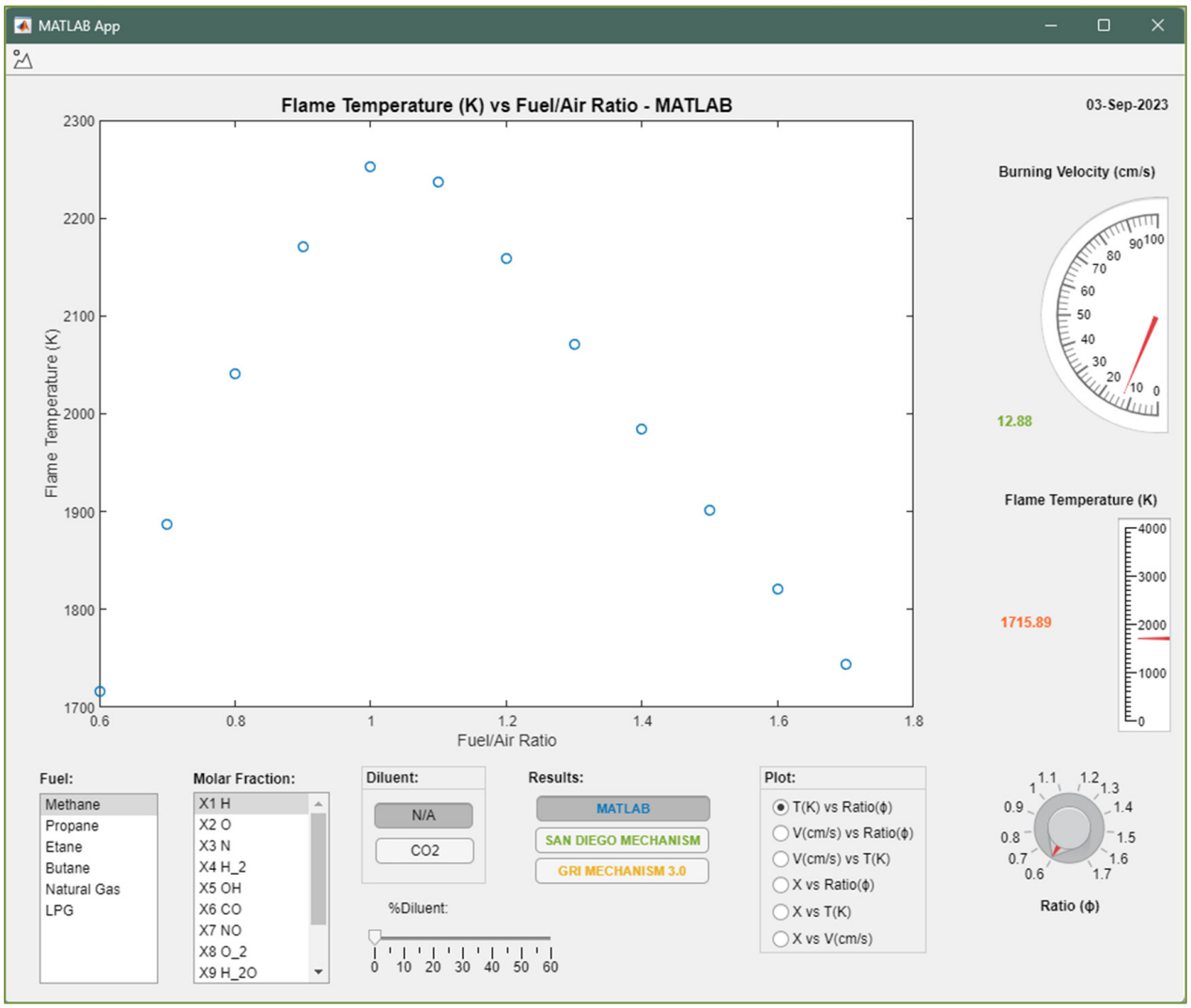


Figure 30. Application first screen with identified tools [8].

Table 1. Tools available in the MATLAB Application [8].

Position	Name of Tool	Description	Selection Type
A	Fuel list	List of fuels available for calculus in the application	Unique
B	Molar fraction list	List of molar fractions available for plotting results	Unique
C	Diluent button list	List of diluents available to use in calculus	Unique
D	Percentage bar of diluent	Percentage of diluent by volume to be considered in fuel	Unique
E	Results button list	Results from resources available to be show in the plot	Multi
F	Plot button list	Type of plot to be show in the screen	Unique
G	Equivalence ratio knob	Knob that allows to see the value of laminar burning velocity and flame temperature in their respective gauges	Unique
H	Laminar burning velocity gauge	Laminar burning velocity value at the knob equivalence ratio value selected	NA
I	Flame temperature gauge	Flame temperature value at the knob equivalence ratio value selected.	NA

4. Results and Discussions

The results obtained by executing the MATLAB Application are presented for three fuels (methane, propane and natural gas). These are compared and a brief discussion of the differences between them is presented for each fuel. Furthermore, the results from the

application and from other relevant bibliography are compared in order to analyze and the behavior of the numerical method.

4.1. Methane

Figure 31 shows the results obtained for methane of flame temperature (a) and laminar burning velocity (b) compared with the research of Andrews and Bradley [11] and Sakhrieh [12].

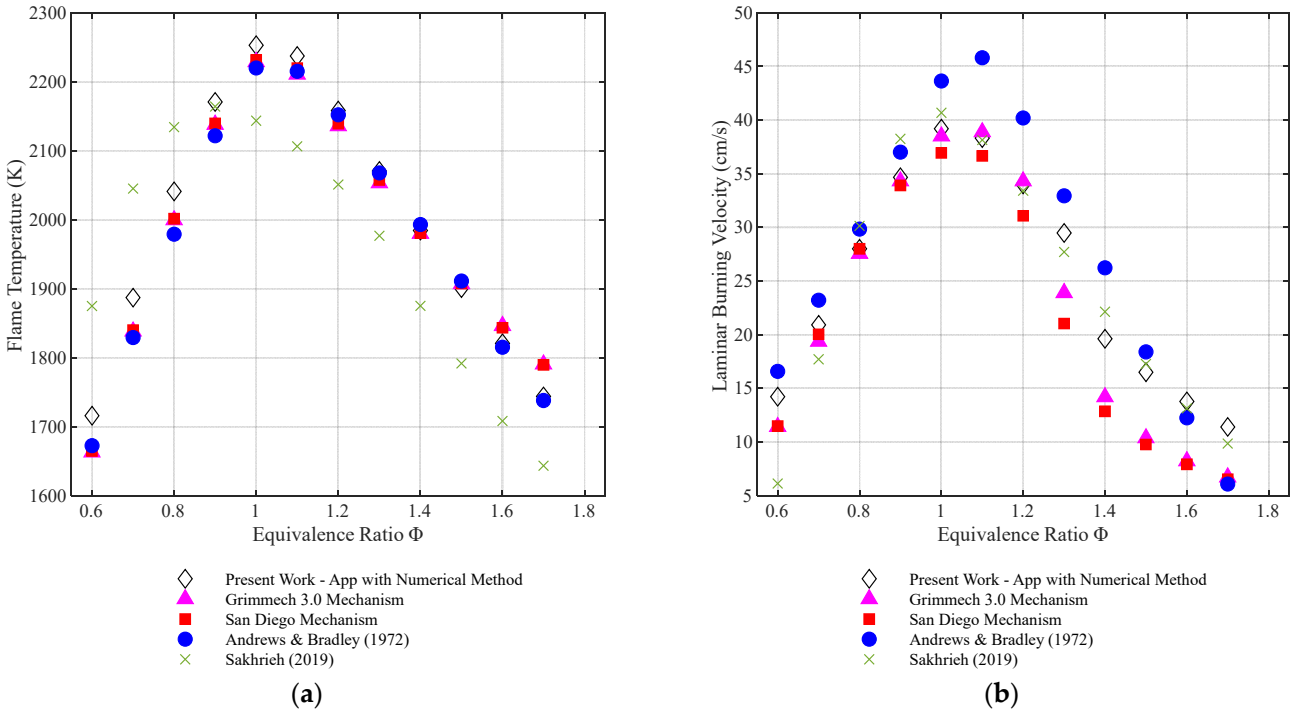


Figure 31. Results comparison with references [11,12] for Methane: (**a**) Flame temperature (K) versus equivalence ratio, (**b**) Laminar burning velocity (cm/s) versus equivalence ratio.

From Figure 31, it is possible to affirm that the numerical method results have the same behavior as the two simulations in Ansys Chemkin (San Diego Mechanism [9] and GRI-MECH 3.0 [10]) and Andrews and Bradley's bomb hot wire method study for flame temperature results [12]. For laminar burning velocity results, the work performed by Sakhrieh [12 is the most accurate one to the numerical method used while Andrews and Bradley's [11] results are higher for mixtures near the stoichiometric.

4.2. Propane

Figure 32 shows the results obtained for propane of flame temperature (a) and laminar burning velocity (b) compared with Jiang, et al.'s research [13] and the investigation performed by Vagelopoulos and Egolfopoulos [14].

Figure 32 evidences that the numerical method flame temperature results have the same behavior as the two simulations in Ansys Chemkin (GRI-MECH 3.0 [10] and San Diego Mechanism [9]) and the numerical results from Jiang et al. [13]. However, in this case, the values for the numerical method using the MATLAB Application are considerably higher than the other resources. In addition, it is possible to observe that propane laminar burning velocity (cm/s) resulting from the numerical method has the same behavior as the other resources except the simulation using the GRI-MECH 3.0 in Ansys Chemkin whose values are higher than all the compared resources.

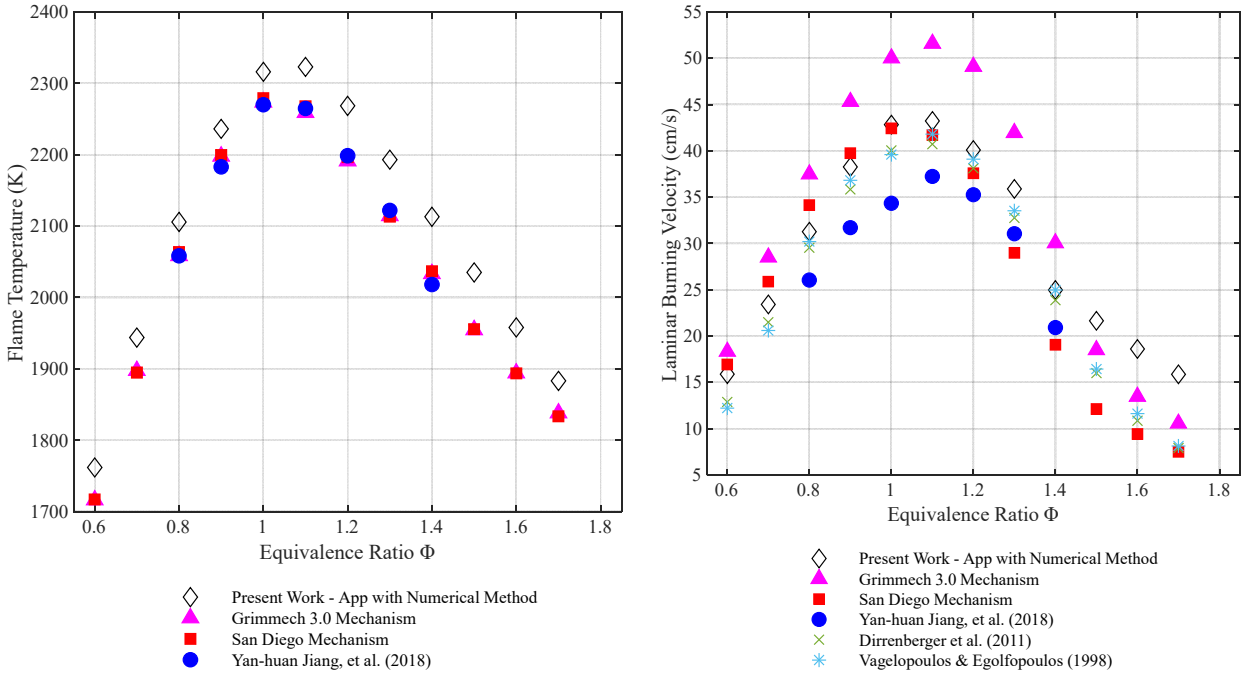


Figure 32. Results comparison for Propane: (**a**) Flame temperature (K) versus equivalence ratio, (**b**) Laminar burning velocity versus equivalence ratio compared with references [13,14].

4.3. Natural Gas

Figure 33 shows for natural gas the laminar burning velocity results from the present investigation (Camisea Natural Gas), Pittsburgh Gas and Indonesian Gas from Dirrenberger [15].

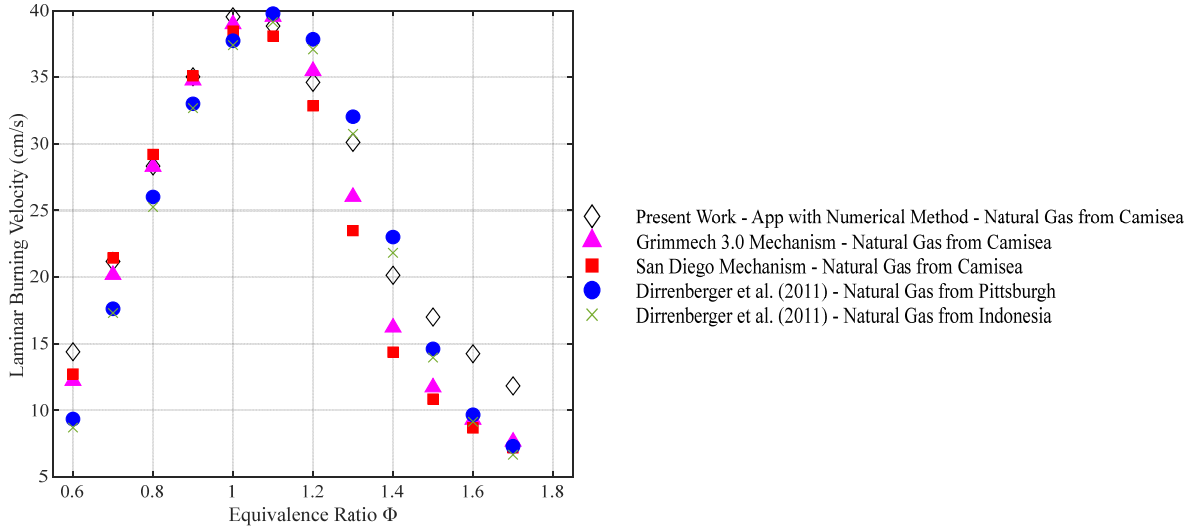


Figure 33. Results comparison with research referenced as [15] for different compositions of Natural Gas Laminar burning velocity versus equivalence ratio.

Figure 33 demonstrates that despite the different compositions of natural gas from the different resources (Camisea, Pittsburgh and Indonesian), the behavior of all the values (numerical method and simulations) are well matched for most of the cases of equivalence ratio, especially for poor and stoichiometric mixtures and near ones.

5. Conclusions

In this work, the numerical method coded in a MATLAB application for the determination of combustion characteristics (products of combustion, the adiabatic flame temperature and the laminar flame velocity) explanation and comparison with other resources was carried out. The following conclusions were reached:

1. In addition to the previously published article referenced as [8], the explanation provided in the present work completes the development plan and implementation of the MATLAB application.
2. The functions and execution sequence are described by using flowcharts and code extracts that allow everybody to understand how it works, and at the same time, the procedures are explicit for each part (New_code.m, Fractions_Derivatives.m, MATLAB_Application.mlapp), in order to provide the necessary information for future improvements.
3. With the MATLAB application, a functional and easy-to-use interface is obtained for the visualization and analysis of the results, whose program code (Code S1, Code S2 and Code S3) is available to modify in order to increase the fuel options (6 fuels by default), add more tools or to improve the whole methodology by adding more combustion products and new equations.
4. The implementation of the procedure for the determination of the laminar burning velocity as one of the novelties of this work is presented and explained as part of the New_code.m description that will allow users to improve the assumptions taken or include any other important variables not considered in the scope of the program.
5. The limitations of the application are: it can be only used for hydrocarbon fuels, which can have or not have oxygen and/or nitrogen; the flame temperature must not exceed 3725 °C and must be lower than 327 °C; and the equivalence ratio must not be too high so it does not allow the formation of free carbon [8].
6. The average execution time obtained by using the functions tic and toc in MATLAB when the application is initialized was 6.48 s, and when the fuel selection is changed (a new calculus is performed for the new selected fuel), it takes 6.18 s.

Supplementary Materials: The following supporting information can be downloaded at: https://doi.org/10.5281/zenodo.7894133, Code S1: New_code.m, File S1: REACTANTS EN THALPY.xlsx; File S2: DILUENTS ENTAHLPY.xlsx; Code S2: Fractions_Derivatives.m; Code S3: MATLAB_Application.mlapp. File S3: Ansys Chemkin Results.xlsx, File S4: Prod ucts.xlsx.

Author Contributions: Conceptualization, F.J.R.; methodology, R.F.C.; software, R.F.C.; validation, F.J.R.; investigation, R.F.C. and F.J.R.; resources, R.F.C.; writing—original draft, R.F.C.; writing—review and editing, F.J.R.; supervision, F.J.R.; project administration, F.J.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research (PI0735) was funded by DFI PUCP.

Data Availability Statement: Not applicable.

Acknowledgments: Thanks are due to the DFI of the Pontificia Universidad Católica del Perú that, through the annual competition for applied research projects CAP PUCP 2021, financed research PI0735.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

F	is the equivalence ratio
AD	is the fraction of the diluent
ID	is the identifier of the selected fuel;
ID2	is the identifier of the selected diluent;

H_Data	is the enthalpy of the fuel;
H_AD	is the enthalpy of the diluent;
n	is the carbon coefficient value of the fuel;
m	is the hydrogen coefficient value of the fuel;
l	is the oxygen coefficient value of the fuel;
k	is the nitrogen coefficient value of the fuel;
AVM	is the average molar mass
Ea	is the activation energy;
HR	is the reactants enthalpy;
H	is the products enthalpy;
der_H_T	is the derivative of enthalpy respect temperature;
T	is the first assumed flame temperature or a temperature iteration;
V	is the laminar burning velocity
X	is the matrix of combustion products
x_1	is the molar fraction of Hydrogen (H) in the products;
x_2	is the molar fraction of Oxygen (O) in the products;
x_3	is the molar fraction of Nitrogen (N) in the products;
x_4	is the molar fraction of Hydrogen (H_2) in the products;
x_5	is the molar fraction of Hydroxide (OH) in the products;
x_6	is the molar fraction of Carbon monoxide (CO) in the products;
x_7	is the molar fraction of Nitric oxide (NO) in the products;
x_8	is the molar fraction of Oxygen (O_2) in the products;
x_9	is the molar fraction of Dihydrogen oxide (H_2O) in the products;
x_{10}	is the molar fraction of Carbon dioxide (CO_2) in the products;
x_{11}	is the molar fraction of Nitrogen (N_2) in the products;
x_{12}	is the molar fraction of Argon (Ar) in the products;
x_{13}	is the number of moles from fuel that gives 1 mol of products;
P	is the pressure;
Kp	is the equilibrium constants at constant pressure;
Ta	is the temperature for adjusting Kp;
PAR	first estimated value of x_{13};
f	is the general linear function composed by 4 constants (F1, F2, F3 and F4);
F1	is the first constant of f equation;
F2	is the second constant of f equation;
F3	is the third constant of f equation;
F4	is the fourth constant of f equation;
df	is the differential equation of f;
ox	is the first estimate value of x_8
fox	is the result of using ox as input in f;
dfox	is the result of using ox as input in df;
B_j	is the matrix equations system of molar products;
new iteration	is the variable that determines if a new iteration is required;
SM	is the molar mass dataset;
HT	is the enthalpy dataset;
CT	is the specific heat capacity dataset;
DCT	is the matrix of derivatives respect temperature;
DCP	is the matrix of derivatives respect pressure;
DD4F	is the matrix of derivatives respect ratio;
C1	is the equation matrix system formed by the derivatives;
Tk	is the temperature used as input for calculate the specific enthalpy of a product;
IT	is the fixed temperature to 0 decimals;
FR	is the fraction part of the temperature;
SH	is the matrix of specific enthalpy of each product;
CP	is the constant pressure heat;
Label2	is the label that shows current date;

app.AD	is the value of diluent in the app;
app.ShowMATLAB	is the variable that shows MATLAB numerical method results;
app.ShowSANDIEGO	is the variable that shows Ansys Chemkin using San Diego mechanism results;
app.ShowGRIMMECH	is the variable that shows Ansys Chemkin using Grim 3.0 mechanism results;
app.PleaseselectaresultLabel	is the variable that shows a message requiring the user to select a type of plot to shown on the application;
app.RatioKnob	is the value set of the ratio knob

Appendix A

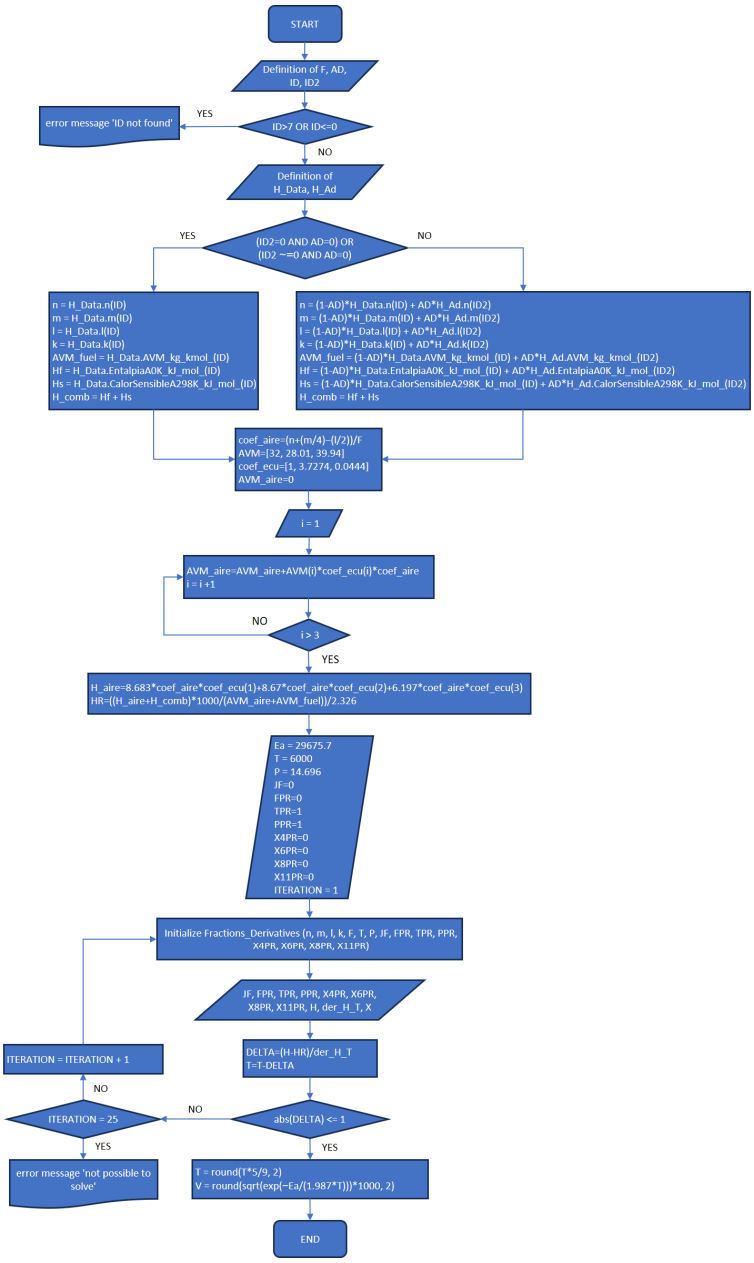
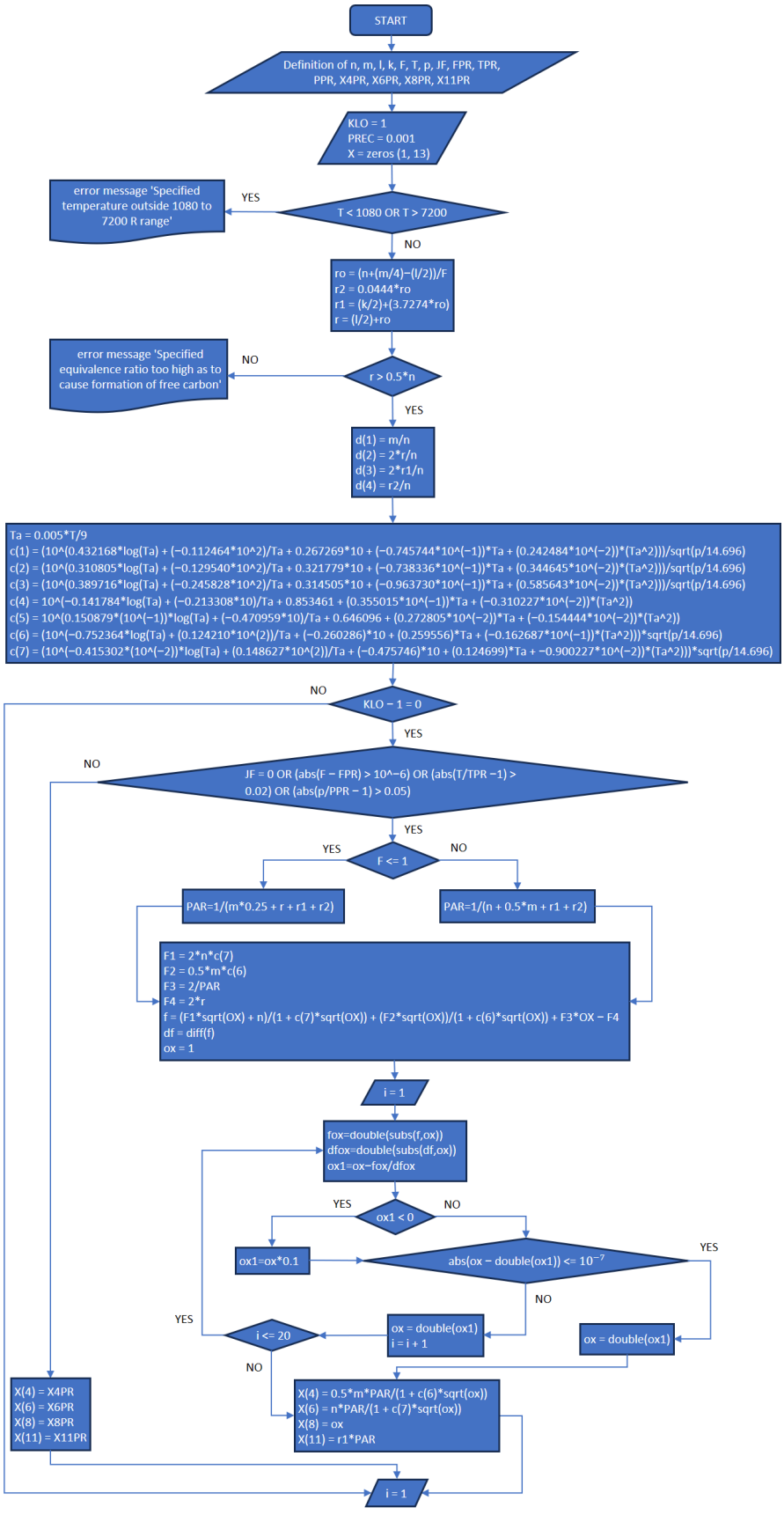
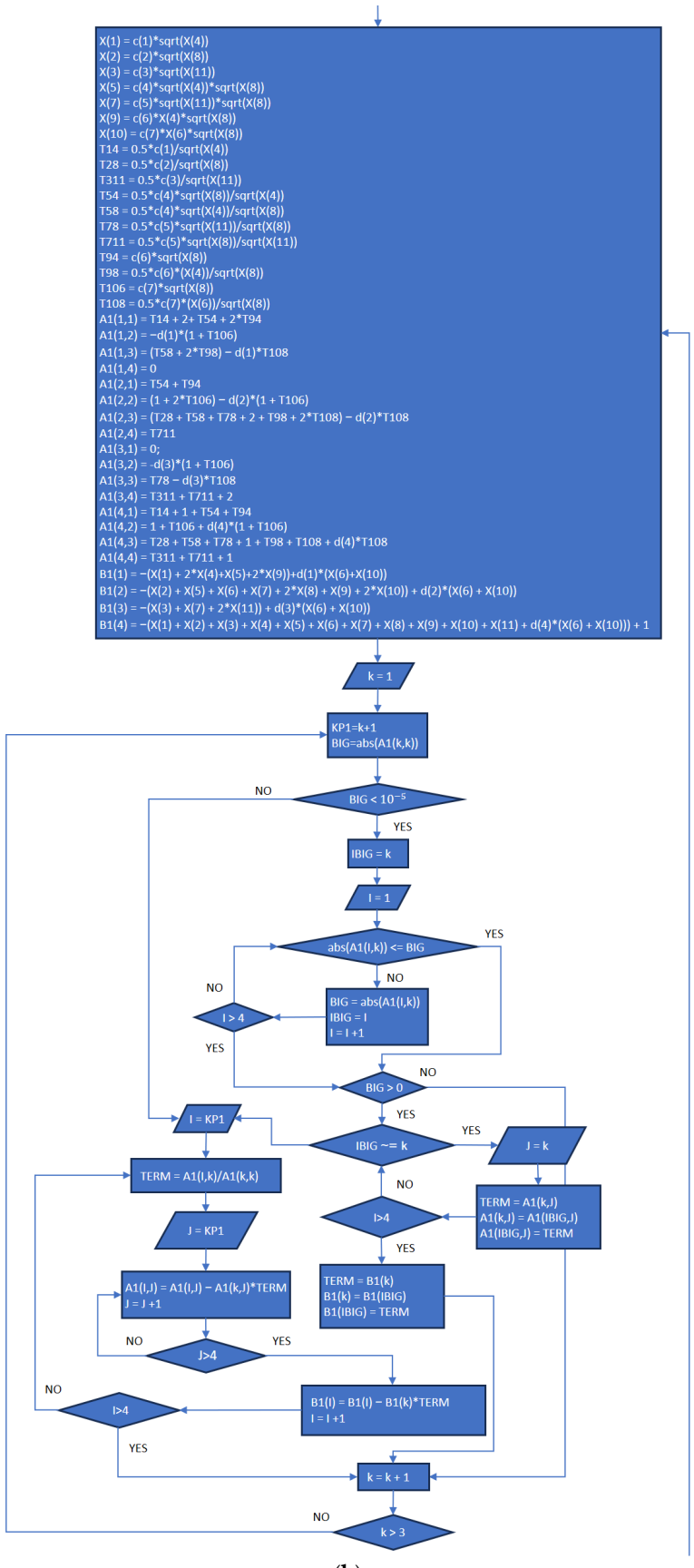


Figure A1. New_code.m flowchart.



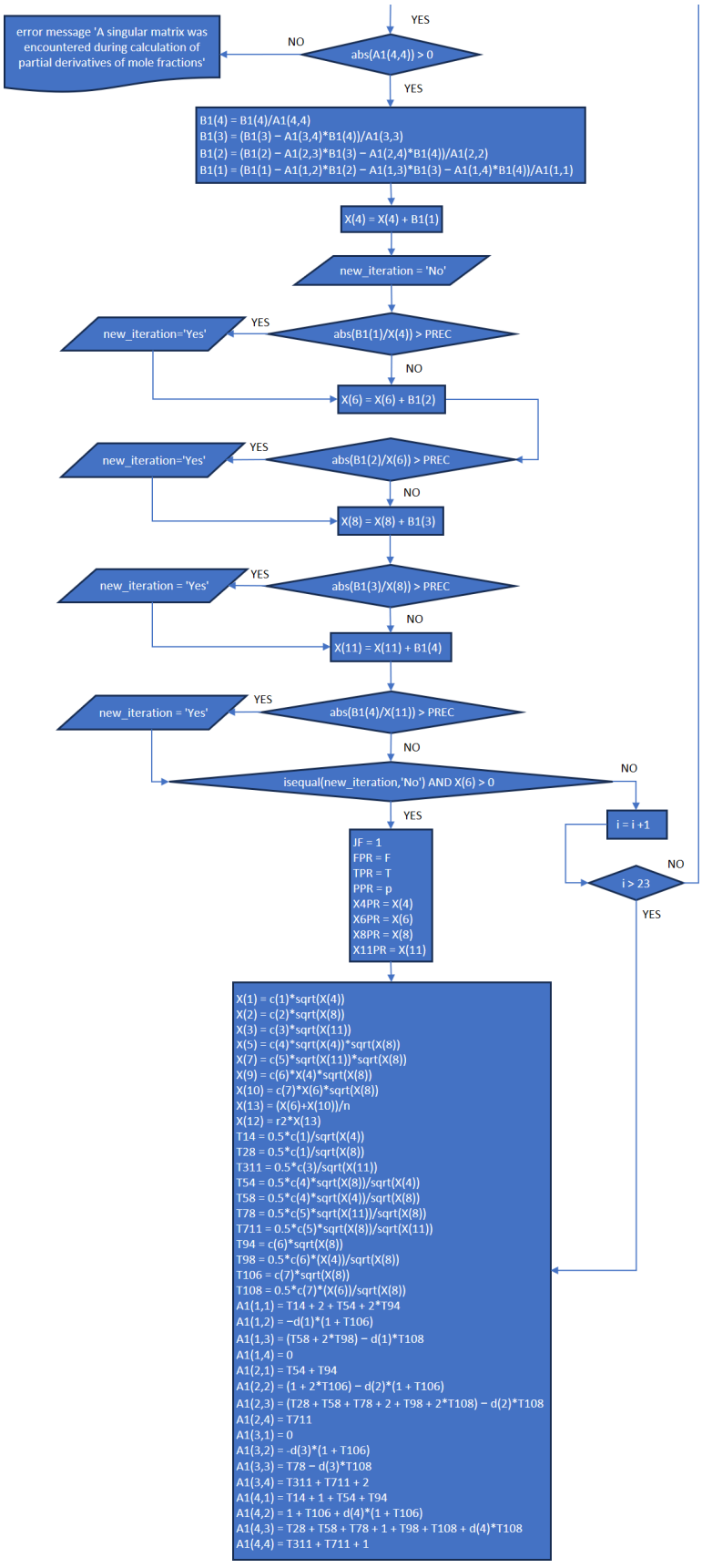
(a)

Figure A2. *Cont.*



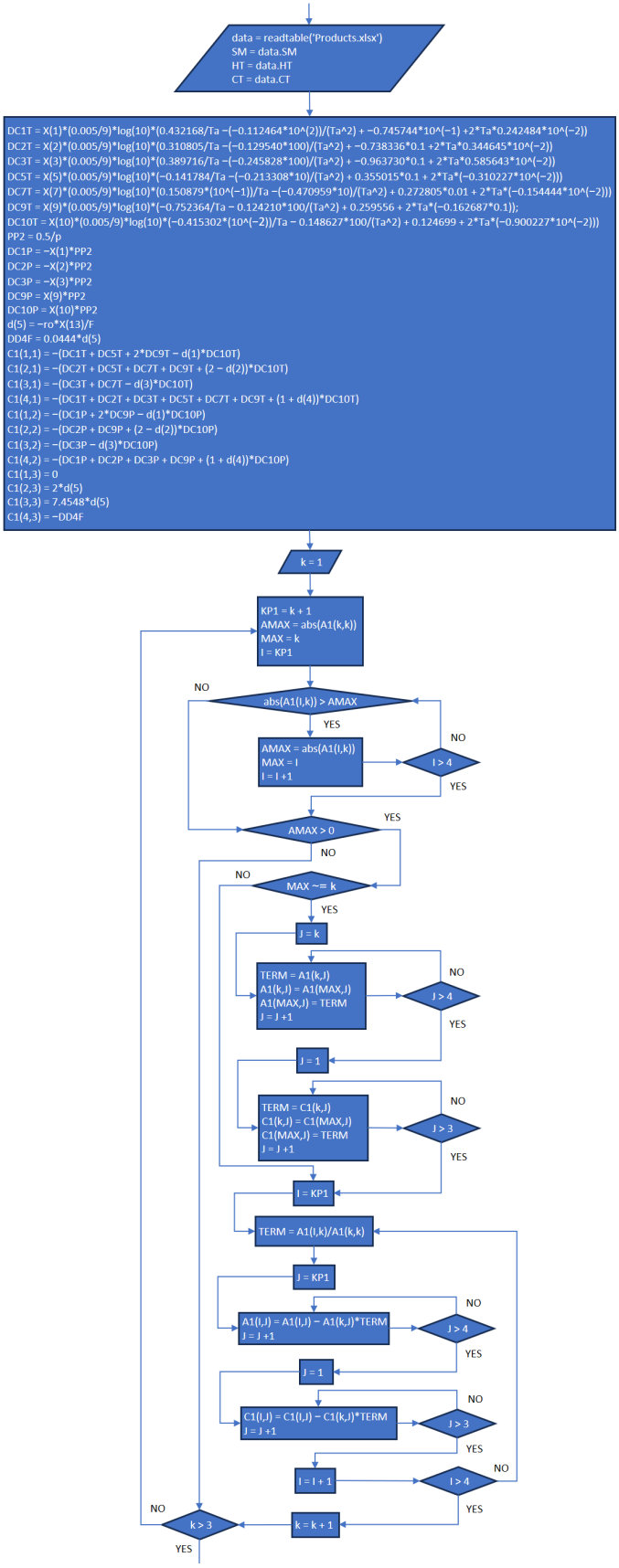
(**b**)

Figure A2. *Cont.*



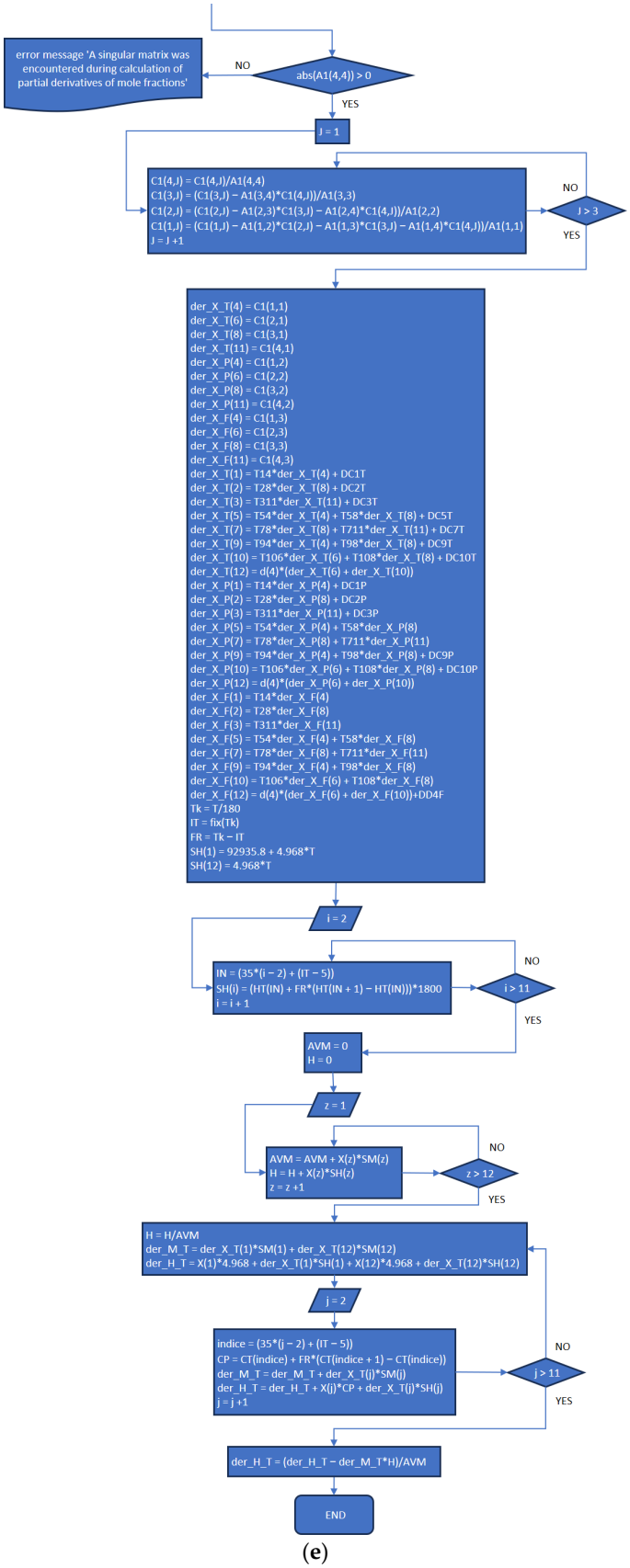
(**c**)

Figure A2. *Cont.*



(d)

Figure A2. *Cont.*



(**e**)

Figure A2. Fractions_derivatives flowchart: (**a**) Part 1, (**b**), Part 2, (**c**) Part 3, (**d**) Part 4, (**e**) Part 5.

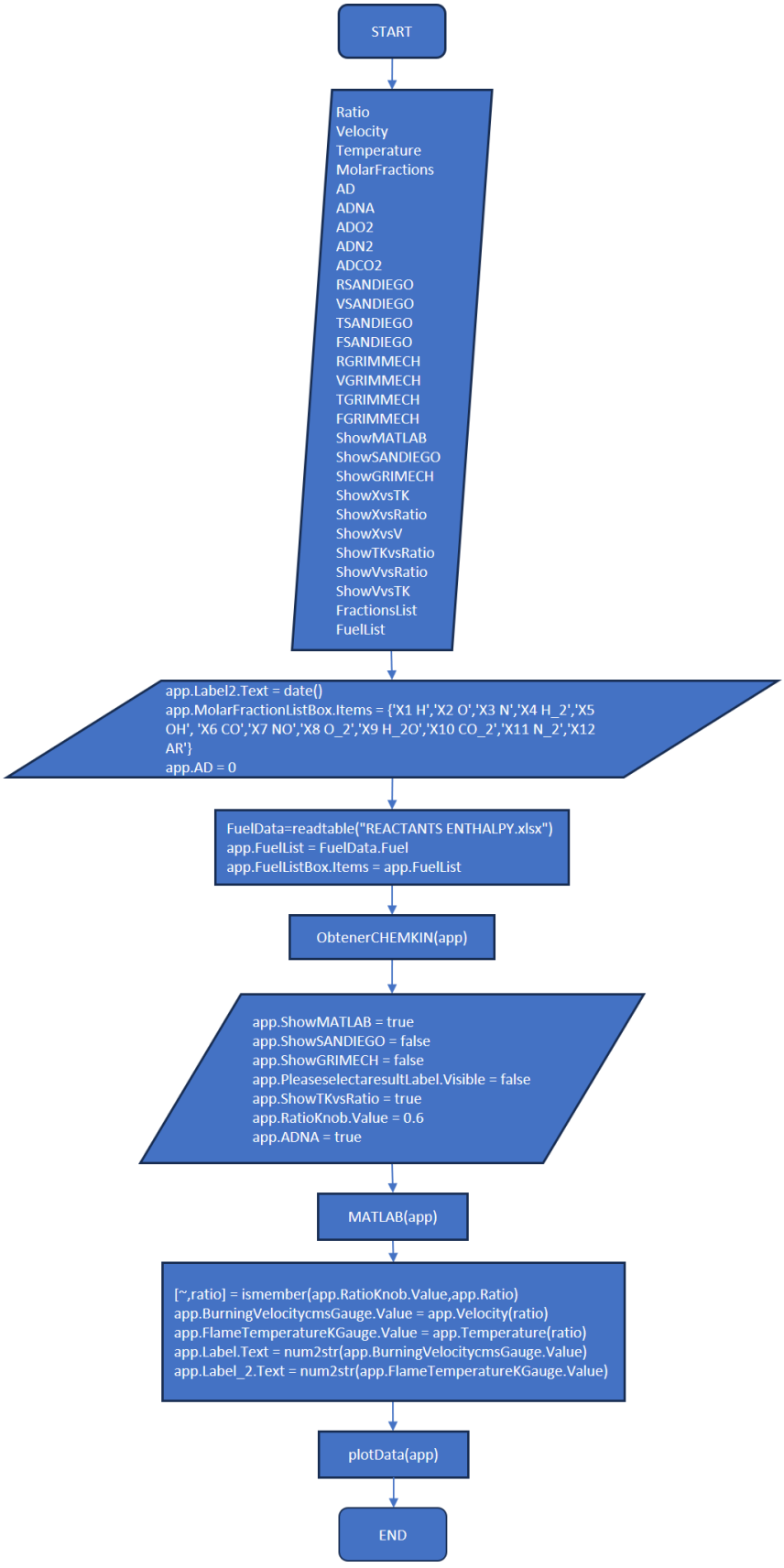


Figure A3. MATLAB_Application.mlapp initialize flowchart.

References

1. Zhu, D.L.; Egolfopoulos, F.N.; Law, C.K. Experimental and numerical determination of laminar flame speeds of methane/(Ar, N_2, CO_2)-air mixtures as function of stoichiometry, pressure, and flame temperature. *Symp. (Int.) Combust.* **1989**, *22*, 1537–1545. [CrossRef]
2. Qin, W.; Egolfopoulos, F.N.; Tsotsis, T.T. Fundamental and environmental aspects of landfill gas utilization for power generation. *Chem. Eng. J.* **2001**, *82*, 157–172. [CrossRef]
3. Hu, E.; Fu, J.; Pan, L.; Jiang, X.; Huang, Z.; Zhang, Y. Experimental and numerical study on the effect of composition on laminar burning velocities of $H_2/CO/N_2/CO_2$/air mixtures. *Int. J. Hydrogen Energy* **2012**, *37*, 18509–18519. [CrossRef]
4. Xiang, L.; Chu, H.; Ren, F.; Gu, M. Numerical analysis of the effect of CO_2 on combustion characteristics of laminar premixed methane/air flames. *J. Energy Inst.* **2019**, *92*, 1487–1501. [CrossRef]
5. Xie, M.; Fu, J.; Zhang, Y.; Shu, J.; Ma, Y.; Liu, J.; Zeng, D. Numerical analysis on the effects of CO_2 dilution on the laminar burning velocity of premixed methane/air flame with elevated initial temperature and pressure. *Fuel* **2020**, *264*, 116858. [CrossRef]
6. Ansys, Inc. *Chemkin-Pro 19.0 Theory Manual*; Ansys, Inc.: Canonsburg, PA, USA, 2018.
7. Olikara, C.; Borman, G.L. A Computer Program for Calculating Properties of Equilibrium Combustion Products with Some Applications to I. C. Engines. In Proceedings of the 1975 Automotive Engineering Congress and Exposition, SAE International, Detroit, MI, USA, 24–28 February 1975. [CrossRef]
8. Cisneros, R.F.; Rojas, F.J. Determination of 12 Combustion Products, Flame Temperature and Laminar Burning Velocity of Saudi LPG Using Numerical Methods Coded in a MATLAB Application. *Energies* **2023**, *16*, 4688. [CrossRef]
9. Combustion, R. San Diego Mechanism Web Page, Chemical-Kinetic Mechanisms for Combustion Applications, San Diego Mechanism Web Page. Available online: http://combustion.ucsd.edu (accessed on 8 June 2023).
10. GRI-MECH 3.0. Available online: http://www.me.berkeley.edu/gri_mech/ (accessed on 8 June 2023).
11. Bradley, D.; Hundy, G.F. Burning velocities of methane-air mixtures using hot-wire anemometers in closed-vessel explosions. *Symp. (Int.) Combust.* **1971**, *13*, 575–583. [CrossRef]
12. Sakhrieh, A. The adiabatic flame temperature and laminar flame speed of methane premixed flames at varying pressures. *Acta Period. Technol.* **2019**, *50*, 220–227. [CrossRef]
13. Jiang, Y.-H.; Li, G.-X.; Li, H.-M.; Li, L.; Tian, L.-L.; Huang, H.-T. Experimental and Numerical Study on the Combustion Characteristics of Propane/Air Laminar Premixed Flame at Elevated Pressure. *Energy Fuels* **2018**, *32*, 9898–9907. [CrossRef]
14. Vagelopoulos, C.M.; Egolfopoulos, F.N. Direct experimental determination of laminar flame speeds. *Symp. (Int.) Combust.* **1998**, *27*, 513–519. [CrossRef]
15. Dirrenberger, P.; Le Gall, H.; Bounaceur, R.; Herbinet, O.; Glaude, P.; Konnov, A.; Battin-Leclerc, F. Measurements of Laminar Flame Velocity for Components of Natural Gas. *Energy Fuels* **2011**, *25*, 3875–3884. [CrossRef]