SSRG International Journal of Recent Engineering Science
ISSN: 2349–7157 / https://doi.org/10.14445/23497157/IJRES-V8I3P103

Volume 8 Issue 3, 12-22, May-Jun, 2021

Original Article

Newton's Method Cubic Equation of State C++ Source Code for Iterative Volume Computation

Abdulhalim Musa Abubakar[1], Ahmad Abubakar Mustapha[2]

[1]*Department of Chemical Engineering, Modibbo Adama University of Science and Technology (MAUTECH), Girei LGA, Yola, Nigeria*
[2]*Department of Computer Science & Engineering, Vigan's Foundation for Science, Technology & Research (VFSTR) (Deemed to be University), Vadlamudi, Guntur, A.P., India*

Received: 25 April 2021 Revised: 26 May 2021 Accepted: 11 June 2021 Published: 24 June 2021

Abstract *- This write-up enumerates the basic cubic equations of state used for pressure, volume and temperature determination for pure substances. It, however, focuses on developing a Dev C++ program for molar volume computation. Since molar volume in all cubic equations of state appears unsolvable due to their nonlinear nature, numerical solution methods, one of which is the Newton-Raphson method, can be utilized to estimate the root of such polynomial equations. The 257 lines of code running from Figure 1 to Figure 6, if executed, will return the molar volume to some certain iteration over a tolerance error specified. The program is a 4-in-1 calculator. The user is capable of working with a desired equation of state at a time. Users should note that the source code converges for either large or small errors specified.*

Keywords - *C++ program, Cubic equations of state, Equation of state, Newton's method, Iteration.*

1. Introduction

An Equation of State (EOS) is a semi-empirical functional relationship between the pressure (P), volume (V) and temperature (T) of a pure substance (Sahay, Edison and Mohamed). The simplest EOS is the ideal gas law itself. As per EOS, every new equation is theoretically believed to be superior to those that precede it. One equation is best for calculating density but not for vapor pressure, while another is very accurate for predicting vapor pressure but not for density. There are different types of EOS that fall into three categories (Mansour): a) First-class EOS are basically cubic equations of state. The cubic equations of state such as the Van der Waals, Redlich and Kwong, Soave-Redlich-Kwong, and Peng-Robinson equations give reasonable results for the thermodynamic behavior of real fluids; b) Second class EOS are non-cubic. They provide accurate results for both vapor and liquid phases, of which the Benedict et al. equation is a good example for this class of equations; and c) Third class EOS which are non-analytical EOS that are highly constrained for some specific fluids. These equations of states, especially those requiring iteration to determine a given parameter, can be difficult to handle. Numerical analysis can be applied here, and one such method is Newton's Method. The method starts by taking an initial value or guesses to get a new approximation to the root. These other new values would then serve as initial to get the next and so on. However, doing this manually might take time, especially if it is not converging quickly. A program can be written to forever prevent these grueling molar volume calculations. An example of such a programming languages is C++.

The program did not, however, include a write-up that evaluates P, T and the number of moles (n) for the cubic EOS, as these parameters don't require iterations. Where molar volume is substituted with (V/n), the cubic EOS, an equation showing the number of moles (n) with up to degree 3 is seen. Such an equation, too, requires iteration. The temperature under Redlich-Kwong EOS would as well require iterations. Deiters et al. (2014) wrote in "Calculation of Densities from Cubic Equations of State: Revisited" but did not, however, capture the programming aspect of finding the densities or molar volume of the EOS.

2. Comparison of cubic EOS

Cubic equations of state (CEOS) are widely used in phase-equilibrium calculations because of their simplicity and accuracy (Soedarto). The cubic equations of state (CEOS) such as Van der Waals, Redlich-Kwong, Soave, and Peng-Robinson are simple models that have been widely used in the oil industry and also to describe the state of reservoir fluids at given conditions (Mansour). Table 1 displays the four CEOS and their constant parameter expressions (Himmelblau and Riggs):

Table 1. Cubic equations of state

Van der Waals EOS $\left(P+\frac{a}{\hat{V}^2}\right)\left(\hat{V}-b\right)=RT$ where $a=\frac{27}{64}\frac{R^2{T_c}^2}{P_c}$ and $b=\frac{RT_c}{8P_c}$	**Redlich-Kwong EOS** $\left[P+\frac{a}{T^{\frac{1}{2}}\hat{V}\left(\hat{V}+b\right)}\right]\left(\hat{V}-b\right)=RT$ where $a=0.42748\frac{R^2{T_c}^{2.5}}{P_c}$ and $b=0.08664\frac{RT_c}{P_c}$
Soave Redlich-Kwong EOS $P=\frac{RT}{\hat{V}-b}-\frac{a\,\alpha}{\hat{V}\left(\hat{V}+b\right)}$ where $a=0.42747\frac{R^2{T_c}^2}{P_c}$; $b=0.08664\frac{RT_c}{P_c}$ and $T_r=\frac{T}{T_c}$ $\alpha=\left[1+(0.48508+1.55171w-0.15613w^2)\left(1-{T_r}^{0.5}\right)\right]^2$	**Peng-Robinson EOS** $P=\frac{RT}{\hat{V}-b}-\frac{a\,\alpha}{\hat{V}\left(\hat{V}+b\right)+b\left(\hat{V}-b\right)}$ where $a=0.45724\frac{R^2{T_c}^2}{P_c}$ and $b=0.07780\frac{RT_c}{P_c}$ $\alpha=\left[1+k\left(1-T_r^{\frac{1}{2}}\right)\right]^2$ where $T_r=\frac{T}{T_c}$ $k=0.37464+1.54226w-0.26992w^2$

The van der Waals EOS is the basis of various EOS. It is not capable of predicting phase equilibrium accurately. Redlich-Kwong EOS needs critical temperature and pressure for P, V or T calculations. This equation is not accurate enough for density and phase-equilibria calculations, including vapor pressures and solubility of solids in supercritical fluids (Soedarto).

The Redlich Kwong EOS can satisfactorily be applied for fugacity, enthalpy and entropy departure calculations as well as gas phase properties. This equation poorly predicts liquid phase properties (Sahay, Edison and Mohamed). The Soave-Redlich-Kwong (SRK) and Peng Robinson EOS are the most applicable in the petroleum industry (Oliveira, Ribeiro and Queimada). Soave replaced the term $T^{-0.5}$ of the RK equation of state by a function α involving temperature and acentric factor. The parameter was formulated to make the equation fit the vapor pressure data of hydrocarbons (Housam and Zakia).

The Peng-Robinson EOS was suggested to satisfy the following objectives (Housam and Nasri, Applications of the peng-robinson equation of state using MATLAB): a) parameters of this EOS should be defined in terms of the critical properties and the acentric factor; b) reasonable accuracy near the critical point, particularly for calculations of the compressibility factor and liquid density; c) a single binary interaction parameter, which should be independent of temperature, pressure and composition, is needed for the mixing rules; and d) PENG EOS should be applicable in natural gas processes.

The PENG EOS provided results similar to the SRK EOS, although it is generally superior in estimating the liquid densities of many materials, especially nonpolar ones. The applicability of the Peng Robinson equation of state in the computation of thermodynamic interactions of volatile organic compounds and biodiesel has also been tested by Sahay et al. (2013).

Working with this set of equations might sometimes be a tedious task, especially if the user is after the molar volume ($\hat{V}$) given other conditions. The term "cubic" in CEOS earns its name because they result in a polynomial of degree 3 in volume if manipulated. These types of nonlinear equations can be solved by applying Numerical Methods like Secant, Fixed-point iteration, Regula-Falsi, Bisection or Newton-Raphson Method.

3. Application of Newton-Raphson Method

This Journal hand-picked the Newton-Raphson analytical method because of its rapid convergence to the root as one reason. By Newton's Method, if f is differentiable and x is an approximate solution of equation f(x) = 0, then a better approximation might be obtained using $\boxed{x_{i+1}=x_i-\frac{f(x_i)}{f'(x_i)}}$, for $i=0,1,2,...$, where i = number of iterations or repetitions (Mate).

EOS in Table 1 can be rewritten in the following way to compute molar volume ($\hat{V}$) of a given gas:

3.1. Van der Waals EOS

Writing the Van der Waals equation as a function of molar volume:

$$f(\hat{V}) = \hat{V}^3 - \left(b + \frac{RT}{P}\right)\hat{V}^2 + \left(\frac{a}{P}\right)\hat{V} - \frac{ab}{P} = 0 \quad (1)$$

$$f'(\hat{V}) = 3\hat{V}^2 - 2\left(b + \frac{RT}{P}\right)\hat{V} + \frac{a}{P} \quad 2)$$

3.2. Redlich-Kwong EOS

The Redlich-Kwong EOS written as function of volume, $f(\hat{V})$ is as follows

$$f(\hat{V}) = \hat{V}^3 - \left(\frac{RT}{P}\right)\hat{V}^2 - \left(b^2 + \frac{RTb}{P} - \frac{a}{P\,T^{\frac{1}{2}}}\right)\hat{V} - \frac{ab}{P\,T^{\frac{1}{2}}} = 0 \quad (3)$$

$$f'(\hat{V}) = 3\hat{V}^2 - \left(\frac{2RT}{P}\right)\hat{V} - \left(b^2 + \frac{RTb}{P} - \frac{a}{P\,T^{\frac{1}{2}}}\right) \quad (4)$$

3.3. Soave Redlich-Kwong EOS

SRK EOS can be presented in the following form:

$$f(\hat{V}) = \hat{V}^3 - \left(\frac{RT}{P}\right)\hat{V}^2 - \left(b^2 + \frac{RTb}{P} - \frac{a\,\alpha}{P}\right)\hat{V} - \frac{ab\alpha}{P} = 0 \quad (5)$$

$$f'(\hat{V}) = 3\hat{V}^2 - \left(\frac{2RT}{P}\right)\hat{V} - \left(b^2 + \frac{RTb}{P} - \frac{a\,\alpha}{P}\right) \quad (6)$$

3.4. Peng-Robinson

Rewriting Peng-Robinson EOS:

$$f(\hat{V}) = \hat{V}^3 + \left(b - \frac{RT}{P}\right)\hat{V}^2 - \left(3b^2 + \frac{2RTb}{P} - \frac{a\,\alpha}{P}\right)\hat{V} + \frac{b^3 + RTb^2 - ab\alpha}{P} = 0 \quad (7)$$

$$f'(\hat{V}) = 3\hat{V}^2 + 2\left(b - \frac{RT}{P}\right)\hat{V} - \left(3b^2 + \frac{2RTb}{P} - \frac{a\,\alpha}{P}\right) \quad (8)$$

For all the EOS, we have:

$$\hat{V}_{i+1} = \hat{V}_i - \frac{f(\hat{V}_i)}{f'(\hat{V}_i)} \quad (9)$$

There is always a starting value (say $\hat{V}_0$) to help calculate the root of equations (1), (3), (5) and (7). For a given problem, guess the volume calculated from ideal gas law as the initial value (that is $\hat{V}_{ideal} = \frac{RT}{P}$). Going the manual way by paper and calculator is quite time-consuming and makes one prone to error. In some situations, available online web calculators for any of the EOS might help, but one might have to grapple with unit conversion as there is hardly an online web calculator that makes provision for selecting different units.

The easiest or fastest of all is the use of MS Excel. Limitation common to all the aforementioned solution approaches is that one has to face the same stress for every new problem encountered/given. Therefore, this write-up wishes to present a multiple-option selection C++ program that computes molar volume, $\hat{V}$, by Newton's iterations for the CEOS.

4. C++ Source Code for Molar Volume Evaluation

The C++ language is a better version of the C language. One of its main features is that it supports object-oriented programming (OOP). C++ is used for scientific computing as well as system programming (Ghosh).

The below C++ syntax uses a do-while statement to solve for numerical solutions or roots of nonlinear equations by Newton's method:

CEOS Volume Iteration.cpp

```
1   /* Iterative Volume Computation by Newton's Method for Cubic EOS */
2   #include <iostream>
3   #include <math.h>
4   #include <iomanip>
5   using namespace std;
6   // VANDER WAALS EOS
7   #define f(V) V*V*V-SE1*V*V+(aVDW/P)*V-(aVDW*bVDW)/P
8   #define d(V) 3*V*V-2*SE1*V+aVDW/P
9   // REDLICH-KWONG EOS
10  #define g(V) V*V*V-(R*T/P)*V*V-SE2*V-SE3
11  #define h(V) 3*V*V-2*(R*T/P)*V-SE2
12  // SOAVE REDLICH-KWONG EOS
13  #define m(V) V*V*V-(R*T/P)*V*V-SE7*V-SE8
14  #define n(V) 3*V*V-2*(R*T/P)*V-SE7
15  // PENG ROBINSON EOS
16  #define x(V) V*V*V+SE9*V*V-SE13*V+SE14
17  #define y(V) 3*V*V+2*SE9*V-SE13
18  int main()
19  {
20      int option, step=1, N;
21      float R, Tc, Pc, T, P, V_ideal, aVDW, bVDW, SE1, aREDL, bREDL, SE2, SE3;
22      float aSRK, bSRK, Tr, w, SE4, SE5, SE6, alphaSRK, SE7, SE8;
23      float aPENG, bPENG, SE9, SE10, SE11, SE12, alphaPENG, SE13, SE14;
24      float V1, f0, f1, d0, e, g0, g1, h0, m0, m1, n0, x0, x1, y0;
25
26      cout<<"\n"<<"COMPUTING VOLUME FROM THESE CUBIC EQUATION OF STATE"<<"\n"<<endl;
27      cout<<setw(13)<<"(1) "<<"Van der Waals"<<endl;
28      cout<<setw(13)<<"(2) "<<"Redlich-Kwong"<<endl;
29      cout<<setw(13)<<"(3) "<<"Soave Redlich-Kwong"<<endl;
30      cout<<setw(13)<<"(4) "<<"Peng Robinson"<<endl<<endl;
31      cout<<"Choose Option: ";
32      cin>>option;
33      switch(option)
```

Fig. 1 Dev C++ Syntax showing various headers

Figure 1 defines the EOS and their derivatives (lines 6-17), as well as declaring many parameters as floats and int (lines 20-24). Equations 1-8 are defined in lines 6-17. SE1 to SE14 are expressions derived out of the general function, $f(\hat{V})$. It is to further reduce the complexity of the equation.

In C++ programming, 'int' stands for integer. Variables of type 'int' declared in line 20 of Figure 1 are examples. 'Float' is a decimal point number. Variables declared as 'float' are expected to be to some certain decimal places.

Output statements in C++ always begin with the keyword 'cout'. Lines 27-30 are output statements showing selectable options to execute. When the program is run, it requests the user to select an option, either 1, 2, 3, or 4, for any of the EOS (based on the input statement keyword 'cin' of line 32). The user is expected to press either 1, 2, 3 or 4 to pick an equation of state. The options are well-interpreted using different cases in C++. They are Case 1 (line 35; Fig. 2), Case 2 (line 86; Fig. 2), Case 3 (line 138; Fig. 4a) and Case 4 (line 197; Fig. 4b).

C++ is case-sensitive. If a variable is declared using a lowercase alphabet, it should be declared so throughout the coding. Mistakes like that will prevent the program from running. This information is for readers who wish to type and run the source code in this article, either for usage or confirmation. It is worthy of note that almost all line of code in C++ ends with a semi-colon (;). Forgetting to type this will result in an error notification when the program is eventually run.

```
34      {
35          case 1:
36              cout<<"\n\t"<<"Enter the following parameters:"<<endl<<endl;
37              cout<<setw(20)<<"T = ";
38              cin>>T;
39              cout<<setw(20)<<"P = ";
40              cin>>P;
41              cout<<setw(20)<<"Tc = ";
42              cin>>Tc;
43              cout<<setw(20)<<"Pc = ";
44              cin>>Pc;
45              cout<<setw(20)<<"R = ";
46              cin>>R;
47              aVDW=(27*pow(R,2)*pow(Tc,2))/(64*Pc);
48              bVDW=(R*Tc)/(8*Pc);
49              SE1=bVDW+(R*T)/P;
50              cout<<endl<<"*************************"<<endl;
51              cout<<"NEWTON-RAPHSON METHOD";
52              cout<<endl<<"*************************"<<endl;
53              cout<<setprecision(4)<<fixed;
54          /* INPUTS */
55              cout<<"\t"<<"Guess value, V_ideal = ";
56              cin>>V_ideal;
57              cout<<"\t"<<"Specify Tolerable Error: ";
58              cin>>e;
59              cout<<"\t"<<"Maximum Iteration = ";
60              cin>>N;
61              cout<<"\n"<<setw(5)<<"Iteration"<<setw(7)<<"V"<<setw(10)<<"f(V)"<<setw(11)<<"f'(V)";
62                  do
63                  {
64                      f0=f(V_ideal);      // = f(V)
65                      d0=d(V_ideal);      // = f'(V)
66                      if(d0==0.0)
67                      {
68                          cout<<"Mathematical Error";
69                          exit(0);
70                      }
71                      V1=V_ideal-f0/d0;       // Newton's Formula: xn+1=xn-f(xn)/f'(xn)
72                      cout<<"\n"<<setw(5)<<step<<setw(13)<<V1<<setw(9)<<f(V1)<<setw(11)<<d(V1);
73                      step=step+1;            // Iteration incremented by '1'
74                      V_ideal=V1;
75                      if(step>N)
76                      {
77                          cout<<"Not Convergent";
78                          exit(0);
79                      }
80                      f1=f(V1);
81                  }
82                  while(fabs(f1)>e);
83                  cout<<endl<<endl<<"Root is V = "<<V1<<endl;
84              break;
85
86          case 2:
87              cout<<"\n\t"<<"Enter the following parameters:"<<endl<<endl;
88              cout<<setw(20)<<"T = ";
89              cin>>T;
90              cout<<setw(20)<<"P = ";
91              cin>>P;
92              cout<<setw(20)<<"Tc = ";
93              cin>>Tc;
94              cout<<setw(20)<<"Pc = ";
95              cin>>Pc;
96              cout<<setw(20)<<"R = ";
97              cin>>R;
```

Fig. 2 C++ syntax for cases 1 and 2

Figure 2 contains coding of the Van der Waal and Redlich-Kwong EOS as Case 1 and 2, respectively. Line 62-82 is a do-while loop for Newton's iteration. The user is allowed to specify the error, **e**, as well as the number of iterations or steps, N.

```
97          cin>>R;
98          aREDL=(0.42748*pow(R,2)*pow(Tc,2.5))/Pc;
99          bREDL=(0.08664*R*Tc)/Pc;
100         SE2=pow(bREDL,2)+((R*T*bREDL)/P)-aREDL/(P*pow(T,0.5));
101         SE3=(aREDL*bREDL)/(P*pow(T,0.5));
102         cout<<endl<<"************************"<<endl;
103         cout<<"NEWTON-RAPHSON METHOD";
104         cout<<endl<<"************************"<<endl;
105         cout<<setprecision(4)<<fixed;
106       /* INPUTS */
107         cout<<"\t"<<"Guess value, V_ideal = ";
108         cin>>V_ideal;
109         cout<<"\t"<<"Specify Tolerable Error: ";
110         cin>>e;
111         cout<<"\t"<<"Maximum Iteration = ";
112         cin>>N;
113         cout<<"\n"<<setw(5)<<"Iteration"<<setw(7)<<"V"<<setw(10)<<"f(V)"<<setw(11)<<"f'(V)";
114             do
115             {
116                 g0=g(V_ideal);      // = f(V)
117                 h0=h(V_ideal);      // = f'(V)
118                 if(h0==0.0)
119                 {
120                     cout<<"Mathematical Error";
121                     exit(0);
122                 }
123                 V1=V_ideal-g0/h0;       // Newton's Formula: xn+1=xn-f(xn)/f'(xn)
124                 cout<<"\n"<<setw(5)<<step<<setw(13)<<V1<<setw(9)<<g(V1)<<setw(11)<<h(V1);
125                 step=step+1;            // Iteration incremented by '1'
126                 V_ideal=V1;
127                 if(step>N)
128                 {
129                     cout<<"Not Convergent";
130                     exit(0);
131                 }
132                 g1=g(V1);
```

Fig. 3 Syntax for Redlich-Kwong execution

Abbreviations like aVDW, bVDW, simply imply 'a' and 'b' constant parameters for Van der Waals EOS. Similarly, that of Redlich-Kwong are subscripted as REDL, SRK for Soave-Redlich-Kwong and PENG for Peng-Robinson (See Figure 3, lines 98-101). Also, see Figure 1 (lines 21-23). The function in the form of 'pow(x,y)', as seen in lines 47, 98, 100-101, 152, 155-156, 211, 214, 216 and 218-220 in almost all the figures, defines variables, say' x' to the power of 'y'.

The program requests T, P, critical temperature and pressure, T_c, and P_c, molar gas constant, R, as well as an acentric factor, ω, to be entered for program execution (one of such is line 139-151 of Figure 4a). It is left for the user to ensure unit consistency of the properties requested as well as their accuracy. A wrong entry will definitely give a wrong output.

Note that the coding is almost the same for all cases. Differences are in the written equations and parameter expressions (for instance, aSRK in line 152 differs from aPENG in line 211 of Figure 5).

```
133             }
134             while(fabs(g1)>e);
135             cout<<endl<<endl<<"Root is V = "<<V1<<endl;
136         break;
137
138     case 3:
139         cout<<"\n\t"<<"Enter the following parameters:"<<endl<<endl;
140         cout<<setw(20)<<"T = ";
141         cin>>T;
142         cout<<setw(20)<<"P = ";
143         cin>>P;
144         cout<<setw(20)<<"Tc = ";
145         cin>>Tc;
146         cout<<setw(20)<<"Pc = ";
147         cin>>Pc;
148         cout<<setw(20)<<"R = ";
149         cin>>R;
150         cout<<setw(20)<<"w = ";
151         cin>>w;
152         aSRK=(0.42747*pow(R,2)*pow(Tc,2))/Pc;
153         bSRK=(0.08664*R*Tc)/Pc;
154         Tr=T/Tc;
155         SE4=0.48508+1.55171*w-0.15613*pow(w,2);
156         SE5=1-pow(Tr,0.5);
157         SE6=1+SE4*SE5;
158         alphaSRK=pow(SE6,2);
159         SE7=pow(bSRK,2)+(R*T*bSRK)/P-(aSRK*alphaSRK/P);
160         SE8=(aSRK*bSRK*alphaSRK)/P;
161         cout<<endl<<"************************"<<endl;
162         cout<<"NEWTON-RAPHSON METHOD";
163         cout<<endl<<"************************"<<endl;
164         cout<<setprecision(4)<<fixed;
165       /* INPUTS */
166         cout<<"\t"<<"Guess value, V_ideal = ";
```

Fig. 4(a) Syntax for Soave-Redlich-Kwong Execution (Case 3)

The lengthy equations were shorten to a 'short expression' tagged 'SE', see lines 155-157 of Figure 4(a), so as to avoid mistakes in the equation entry that would clearly affect the result accuracy. The precision is set to 4 (i.e. 4 decimal places) in line 164, to be maintained throughout the calculations.

```
167             cin>>V_ideal;
168             cout<<"\t"<<"Specify Tolerable Error: ";
169             cin>>e;
170             cout<<"\t"<<"Maximum Iteration = ";
171             cin>>N;
172             cout<<"\n"<<setw(5)<<"Iteration"<<setw(7)<<"V"<<setw(10)<<"f(V)"<<setw(11)<<"f'(V)";
173                 do
174                 {
175                     m0=m(V_ideal);      // = f(V)
176                     n0=n(V_ideal);      // = f'(V)
177                     if(n0==0.0)
178                     {
179                         cout<<"Mathematical Error";
180                         exit(0);
181                     }
182                     V1=V_ideal-m0/n0;       // Newton's Formula: xn+1=xn-f(xn)/f'(xn)
183                     cout<<"\n"<<setw(5)<<step<<setw(13)<<V1<<setw(9)<<m(V1)<<setw(11)<<n(V1);
184                     step=step+1;            // Iteration incremented by '1'
185                     V_ideal=V1;
186                     if(step>N)
187                     {
188                         cout<<"Not Convergent";
189                         exit(0);
190                     }
191                     m1=m(V1);
192                 }
193                 while(fabs(m1)>e);
194                 cout<<endl<<endl<<"Root is V = "<<V1<<endl;
195             break;
196 
197         case 4:
198             cout<<"\n\t"<<"Enter the following parameters:"<<endl<<endl;
199             cout<<setw(20)<<"T = ";
200             cin>>T;
201             cout<<setw(20)<<"P = ";
```

Fig. 4(b) Continuation of C++ Syntax for SRK Execution (Case 3)

Figure 4b is a continuation of the codes for SRK EOS. The variable' m' and 'n' stands for $f(\hat{V})$ and $f'(\hat{V})$ corresponding to equations (5) and (6), respectively. The Newton expression in terms of these functions can be seen clearly in line 182. Initial guess was taken as V_{ideal} for the functions. Figure 5 has the execution for Peng Robinson EOS:

```
202             cin>>P;
203             cout<<setw(20)<<"Tc = ";
204             cin>>Tc;
205             cout<<setw(20)<<"Pc = ";
206             cin>>Pc;
207             cout<<setw(20)<<"R = ";
208             cin>>R;
209             cout<<setw(20)<<"w = ";
210             cin>>w;
211             aPENG=(0.45724*pow(R,2)*pow(Tc,2))/Pc;
212             bPENG=(0.07780*R*Tc)/Pc;
213             SE9=bPENG-(R*T)/P;
214             SE10=0.37464+1.54226*w-0.26992*pow(w,2);
215             Tr=T/Tc;
216             SE11=1-pow(Tr,0.5);
217             SE12=1+SE10*SE11;
218             alphaPENG=pow(SE12,2);
219             SE13=3*pow(bPENG,2)+(2*R*T*bPENG/P)-(aPENG*alphaPENG/P);
220             SE14=(pow(bPENG,3)+R*T*pow(bPENG,2)-aPENG*bPENG*alphaPENG)/P;
221             cout<<endl<<"*************************"<<endl;
222             cout<<"NEWTON-RAPHSON METHOD";
223             cout<<endl<<"*************************"<<endl;
224             cout<<setprecision(4)<<fixed;
225         /* INPUTS */
226             cout<<"\t"<<"Guess value, V_ideal = ";
227             cin>>V_ideal;
228             cout<<"\t"<<"Specify Tolerable Error: ";
229             cin>>e;
230             cout<<"\t"<<"Maximum Iteration = ";
231             cin>>N;
232             cout<<"\n"<<setw(5)<<"Iteration"<<setw(7)<<"V"<<setw(10)<<"f(V)"<<setw(11)<<"f'(V)";
233                 do
234                 {
235                     x0=x(V_ideal);      // = f(V)
236                     y0=y(V_ideal);      // = f'(V)
```

Fig. 5 C++ syntax for Peng-Robinson (case 4)

In line 215 of Figure 5, the reduced pressure, T_r has been accounted for. The user would not be asked to enter the value of T_r but rather the critical temperature, T_c will be requested as $T_r = \frac{T}{T_c}$. Figure 6 is the remaining the coding:

```
237                     if(y0==0.0)
238                     {
239                         cout<<"Mathematical Error";
240                         exit(0);
241                     }
242                     V1=V_ideal-x0/y0;       // Newton's Formula: xn+1=xn-f(xn)/f'(xn)
243                     cout<<"\n"<<setw(5)<<step<<setw(13)<<V1<<setw(9)<<x(V1)<<setw(11)<<y(V1);
244                     step=step+1;            // Iteration incremented by '1'
245                     V_ideal=V1;
246                     if(step>N)
247                     {
248                         cout<<"Not Convergent";
249                         exit(0);
250                     }
251                     x1=x(V1);
252                 }
253                 while(fabs(x1)>e);
254                 cout<<endl<<endl<<"Root is V = "<<V1<<endl;
255             break;
256         }
257     }
```

Fig. 6 Continuation of syntax for Peng-Robinson execution (case 4)

The keyword break (line 255 of Figure 6) causes the program control to exit from the switch block. The keyword break must be included at the end of each case statement.

5. Program Execution

5.1. Sample Problem

For propane, C_3H_8: $T_c = 205.92°F = 665.92R$, $P_c = 615.5\ psi$ and acentric factor, $\omega = 0.1529$ (Mansour), with number of moles, $n = 1.136\ lbmol$, $R = 10.73\ \frac{psi\ ft^3}{lbmol\ R}$, at temperature, $T = 683R$ and pressure, $P = 679.7\ psi$, calculate the molar volume using the four CEOS. Do this for iteration number N = 10 and error, e = 0.001. Compare the result.

What, then, is the initial guess? The initial guess, as stated earlier, is, $\hat{V} = \frac{RT}{P} = \frac{10.73 \times 683}{679.7} = 10.782\ f^3/mol$. This, together with five other variables, will be requested by the program when run. An additional variable (i.e. acentric factor, ω) will be requested by the program when the user chooses SRK and PENG equations of state.

Table 2. Result showing iteration

1. VAN DER WAALS

```
Choose Option: 1

        Enter the following parameters:

              T = 683
              P = 679.7
             Tc = 665.92
             Pc = 615.5
              R = 10.73

*************************
NEWTON-RAPHSON METHOD
*************************
        Guess value, V_ideal = 10.782
        Specify Tolerable Error: 0.001
        Maximum Iteration = 10

Iteration      V       f(V)       f'(V)
    1        8.4976  93.0404    60.2056
    2        6.9522  27.9755    26.3885
    3        5.8920   8.5002    11.4763
    4        5.1514   2.5796     5.0593
    5        4.6415   0.7048     2.5548
    6        4.3656   0.1077     1.8499
    7        4.3074   0.0027     1.7595
    8        4.3058   0.0000     1.7574

Root is V = 4.3058
```

2. REDLICH_KWONG

```
Choose Option: 2

        Enter the following parameters:

              T = 683
              P = 679.7
             Tc = 665.92
             Pc = 615.5
              R = 10.73

*************************
NEWTON-RAPHSON METHOD
*************************
        Guess value, V_ideal = 10.782
        Specify Tolerable Error: 0.001
        Maximum Iteration = 10

Iteration      V       f(V)       f'(V)
    1        8.3719 111.2589    69.3885
    2        6.7685  32.7285    31.1361
    3        5.7173   9.3609    14.4302
    4        5.0686   2.4075     7.4284
    5        4.7445   0.4306     4.8761
    6        4.6562   0.0262     4.2899
    7        4.6501   0.0001     4.2510

Root is V = 4.6501
```

3. SOAVE-REDLICH-KWONG

```
Choose Option: 3

        Enter the following parameters:

                T = 683
                P = 679.9
               Tc = 665.92
               Pc = 615.5
                R = 10.73
                w = 0.1529

**************************
NEWTON-RAPHSON METHOD
**************************
        Guess value, V_ideal = 10.782
        Specify Tolerable Error: 0.001
        Maximum Iteration = 10

Iteration      V      f(V)      f'(V)
    1       8.3851 110.1346    69.5138
    2       6.8008  32.1103    31.4898
    3       5.7810   8.9460    14.9832
    4       5.1840   2.1273     8.2141
    5       4.9250   0.3027     5.9432
    6       4.8741   0.0102     5.5438
    7       4.8722   0.0000     5.5297

Root is V = 4.8722
```

4. PENG-ROBINSON

```
Choose Option: 4

        Enter the following parameters:

                T = 683
                P = 679.7
               Tc = 665.92
               Pc = 615.5
                R = 10.73
                w = 0.1529

**************************
NEWTON-RAPHSON METHOD
**************************
        Guess value, V_ideal = 10.782
        Specify Tolerable Error: 0.001
        Maximum Iteration = 10

Iteration      V      f(V)      f'(V)
    1       8.2919 123.8681    75.4588
    2       6.6504  35.9876    34.3072
    3       5.6014   9.9288    16.4772
    4       4.9988   2.2958     9.2205
    5       4.7498   0.3018     6.8581
    6       4.7058   0.0084     6.4792
    7       4.7045   0.0000     6.4682

Root is V = 4.7045
```

Table 2 is an amazing display of the program source code output window for each EOS selected. The window has a black background, and the contents are white. The molar volumes obtained are to four decimal places considering N = 10. This program aims to be a CEOS calculator to be applied by Chemistry and Chemical Engineering students of Polytechnics and Universities. The results of Table 2 can always be tested manually using a calculator. At this point, there is a need to define some nomenclature:

Nomenclature

T	Temperature (absolute)
P	Pressure
a, b	Constant parameter
$T_c,\ P_c$	Critical temperature and pressure
R	Molar gas constant
T_r	Reduced temperature
$k,\ \alpha$	Constant parameters
w	Acentric factor
n	No. of moles
V	Volume of fluid
$\hat{V}$	Molar volume $\left(=\frac{V}{n}\right)$
$f(\hat{V})$	Function of $\hat{V}$ = f(V) = g(V) = m(V) = x(V) for VDW, REDL, SRK and PENG EOS, respectively (see Figure 1)
$f'(\hat{V})$	Derivative of $\hat{V}$ = d(V) = h(V) = n(V) = y(V) for VDW, REDL, SRK and PENG EOS, respectively (see Figure 1)

6. Conclusion

The program was written using Dev-Cpp 5.11 TDM-GCC 4.9.2 version. The results are molar volume computations only by Newton's Iterative Method. They exclude temperature, pressure and number of moles determination. They also exclude alternative numerical techniques for such problems. For this program, entering the right number of iterations and percentage error determines the convergent of the initial guess to the desired root.

Users should note that the source code converges for either large or small error values depending on the number of iterations entered. Entering 10 as a number of iterations will not necessarily mean that C++ should display the result up to the 10^{th} iteration. In general, this coding is for CEOS; it excludes other classes of EOS. Similar codes that incorporate other EOS by either C++ or other programming languages can be proposed for further studies.

This work is not discouraging manual calculations. It is, however, simplifying the rigorous calculation steps involved before arriving at a solution. It will serve as a confirmation calculator for students and teachers alike who might encounter CEOS problems in the course of their studies.

Acknowledgement

Avalanche of gratitude goes to Mr. David Asimiakhuini, who mentored us on the basis of C++ Computer Programming at University of Maiduguri in the year 2014-2016. I specifically acknowledge Dr. Abdu Zubairu of the Department of Chemical Engineering, University of Maiduguri, for his lecture on Equations of State in the year 2014 when I was an undergraduate student of the institution. Your immense contributions are worthy of recognition. I throw more weight of recognition to M. Tech. Ahmad Abubakar Mustapha (a friend at VFSTR University, India) is also a C++ expert. Ultimately, glory be to Allah Almighty for granting us the wisdom to put this knowledge into use.

References

[1] Ulrich K. Deiters, and Ricardo Macías-Salinas, "Calculation of Densities from Cubic Equations of State: Revisited," *Industrial & Engineering Chemistry Research*, vol. 53, no. 6, pp. 2529−2536, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[2] Pallab Ghosh, *Numerical Methods with Computer Programs in C++*, PHI Learning Private Limited, 2009. [Google Scholar]

[3] David Mautner Himmelblau, and James B. Riggs, *Basic Principles and Calculations in Chemical Engineering*, New Jersey: Prentice Hall, 1982. [Google Scholar]

[4] Zakia Nasri, and Housam Binous, "Applications of the Soave–Redlich–Kwong Equation of State Using Mathematica," *Journal of Chemical Engineering of Japan*, vol. 40, no. 6, pp. 534–538, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[5] Zakia Nasri, and Housam Binous, "Applications of the Peng-Robinson Equation of State using MATLAB," *Chemical Engineering Education*, vol. 43, no. 2, pp. 115-124, 2009. [Google Scholar] [Publisher Link]

[6] Mohamed Eman Mansour, *Equation of State, Inverse Heat Conduction and Heat Exchangers*, Cairo: Intech Open, 2020. [Google Scholar]

[7] Attila Mate, *Introduction to Numerical Analysis with C Program*, Brooklyn College of the City University of New York, 2014. [Google Scholar] [Publisher Link]

[8] M.B. Oliveira et al., "Modeling Phase Equilibria Relevant to Biodiesel Production: A Comparison of gE Models, Cubic EoS, EoS-gE and Association EoS," *Industrial & Engineering Chemistry and Research*, vol. 50, no. 4, pp. 2348-2358, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[9] Sashay Ramdharee, Edison Muzenda, and Mohamed Belaid, "A Review of the Equations of State and their Applicability in Phase Equilibrium Modeling," *International Conference on Chemical and Environmental Engineering*, pp. 84-87, 2013. [Google Scholar] [Publisher Link]

[10] S H. Ratnawati, "Improvement of the Redlich-Kwong Equation of State by Modification of Co-Volume Parameter," *Reaktor*, vol. 13, no. 1, pp. 58-65, 2010. [Google Scholar] [Publisher Link]