

MA348 Numerical Analysis, Thermodynamics

David Jefts

February 2, 2019

Introduction

The goal of this lab is to use various algorithms to estimate the root of a function. In this case, the function was van der Waal's equation of state, $(P + \frac{a}{v^2})(v - b) = RT$, where P is the Pressure in atmospheres, R is the Gas Constant for oxygen in atmospheres per mole-Kelvin, T is the temperature in Kelvin, v is the modal volume and $v = \frac{V}{n}$, V is the total volume, n is the number of moles of gas present, a is the measure of the average attraction between particles, and b is the volume excluded by a mole of particles. In many chemical engineering models, a very accurate modal volume of an atom or molecule, in this case oxygen, is required in order to properly construct containment apparatuses for these gases. van der Waal's equation of state is an expansion upon the classic Ideal Gas Law formula, $PV = nRT$. Using the relationship $v = \frac{V}{n}$, the Gas Law formula used for this lab is $Pv = RT$. In this lab, the values for R , a , and b are constant and known, T and P are not constant but known, and v changes relative to the previous variables based on van der Waal's equation.

Theory-Analysis

The function for this lab is van der Waal's equation, $(P + \frac{a}{v^2})(v - b) = RT$, and the objective is to estimate roots for this function. v is the changing variable, essentially the ' x ' value, so the function must be solved in terms of v :

$$Pv^3 - (bP + RT)v^2 + av - ab = 0$$

This function serves as the main ' $f(v)$ ' function for the remainder of this report. The Ideal Gas Law formula solved for v is:

$$v = \frac{RT}{P}$$

The only assumptions in this report are the oxygen Gas Constant values for R , a , and b . For this report, $R \approx 0.082054$, $a \approx 1.360$, and $b \approx 0.03183$. Additionally, the Fortran installation used to compute root values is only capable of representing 16 decimal places and is ineffective at representing small numbers.

Numerical Solution

This lab was solved using Fortran code to estimate the roots of the function and gnuplot to plot and tabulate the values. Two methods were used to compute the root values for the function- the Bisection Method and the False Position Method.

The Bisection Method involves determining a range in which there should be a root ($[a, b]$), then finding the value at the midpoint of that range (referred to as x_m and calculated with the function $\{x_m = \frac{a+b}{2}\}$). Each iteration then involves comparing x_m with the function values for $f(a)$ and $f(b)$. If $\{f(a) \times f(x_m) < 0\}$ then the b value is replaced by x_m . If $\{f(a) \times f(x_m) > 0\}$ then the a value is replaced by x_m . This method iteratively moves each edge of the range closer and closer towards the true root value of the function, halving the maximum error each iteration.

The False Position Method is very similar to the Bisection Method, except x_m is calculated with the formula $\frac{[f(b) \times a] - [f(a) \times b]}{f(b) - f(a)}$. This formula is the equivalent of the secant line from a to b , with x_m the point where the secant line crosses the x-axis. In some scenarios this method can work much faster than the Bisection Method.

Results and Discussion

Figure 1 is the plot of the function over the range $[0.5, 2.5]$ and shows that the solution is definitively between 2.0 and 2.5.

Figure 2 in Appendix A plots the error during each iteration of the algorithm function, where each line represents one of the above-mentioned estimation methods, the x-axis is the number of iterations and the y-axis is the error. In this graph scenario $P = 10atm$ and $T = 300K$. This graph seems to indicate that the False Positive Roots Estimation Method both starts with a smaller error and converges towards the desired value faster. The Bisection Method converged to a solution within 52 iterations however the False Positive Method converged in 13 iterations, as shown in the table below.

n	Bisection Error	False Positive Error
1	2.25	2.090412
2	0.125	2.74E-02
3	6.25E-02	1.37E-03
4	3.13E-02	6.75E-05
5	1.56E-02	3.32E-06
6	7.81E-03	1.63E-07
7	3.91E-03	8.04E-09
8	1.95E-03	3.96E-10
9	9.77E-04	1.95E-11
10	4.88E-04	9.57E-13
11	2.44E-04	4.71E-14
12	1.22E-04	2.66E-15
13	6.10E-05	0
14	3.05E-05	
15	1.53E-05	
16	7.63E-06	
17	3.81E-06	
18	1.91E-06	
19	9.54E-07	
20	4.77E-07	
...
45	1.42E-14	
46	7.11E-15	
47	3.55E-15	
48	1.78E-15	
49	8.88E-16	
50	4.44E-16	
51	4.44E-16	
52	0	

Table 1: Table of values for the error on each estimation method

Conclusions

The False Positive Method appears to be much more efficient without sacrificing any error, in the future that may not always be the case though. If this were to be done by hand the most accurate way would be the False Positive Method as the Bisection Method is much slower to converge. There are other ways to optimize this function. The Bisection and False Positive Methods of finding roots are just a two from a plethora of various solvers and algorithms that accomplish the same this.

Appendix A

Important Graphs

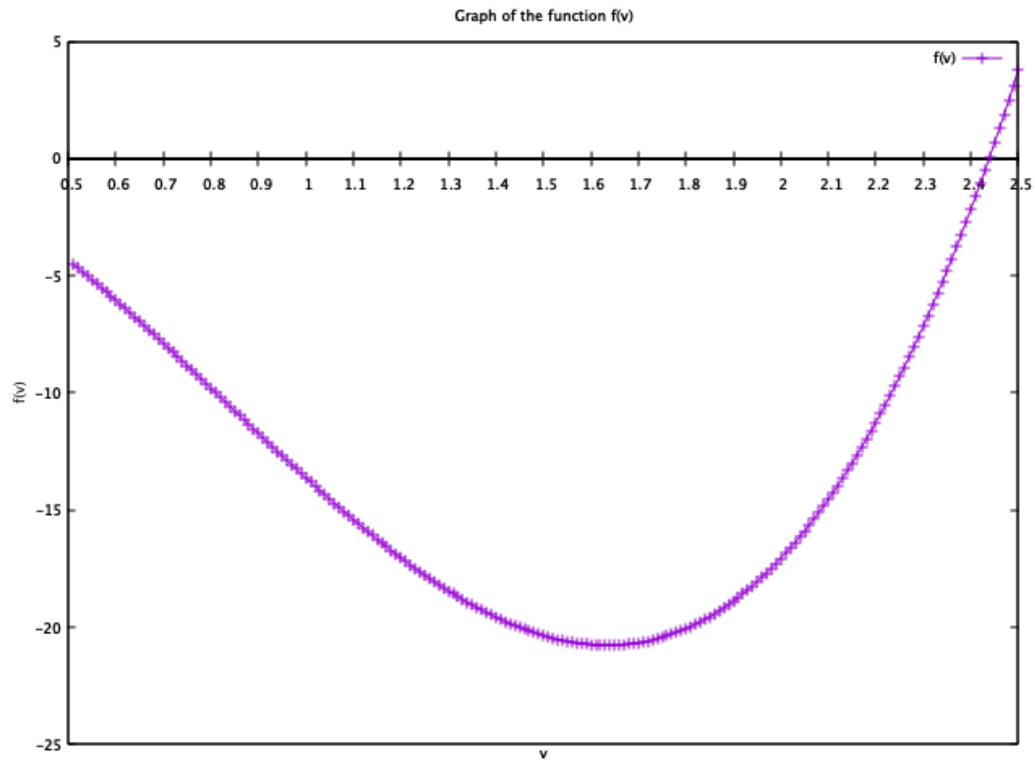


Figure 1: Graph of the $f(v)$ function

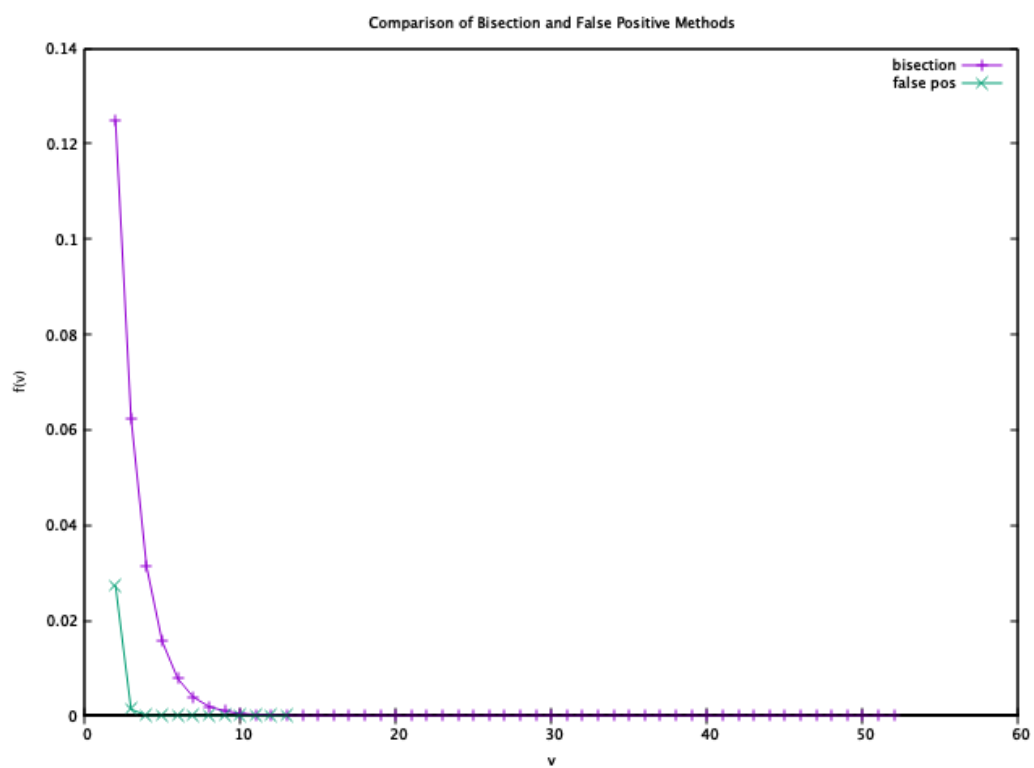


Figure 2: Bisection and False Position Errors when $P = 10atm$ and $T = 300K$

Appendix B

Graphs of each combination of Pressure/Temperature. The green line represents the graph of the Ideal Gas Law on the interval while the purple line represents the False Position Estimation function on the interval. v is the modal volume estimated by the False Position Method.

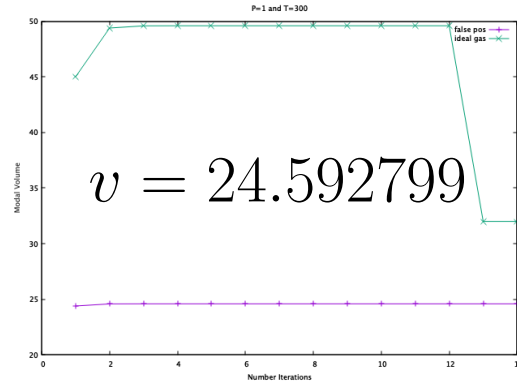


Figure 3: $P = 1atm$, $T = 300K$

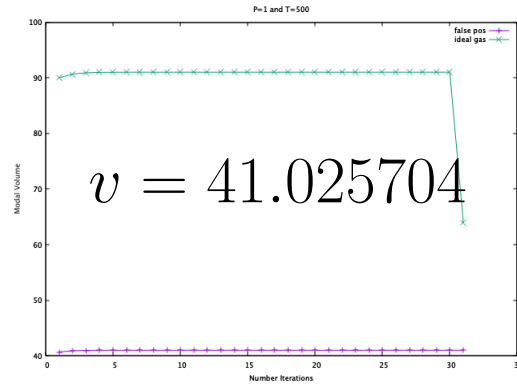


Figure 4: $P = 1atm$, $T = 500K$

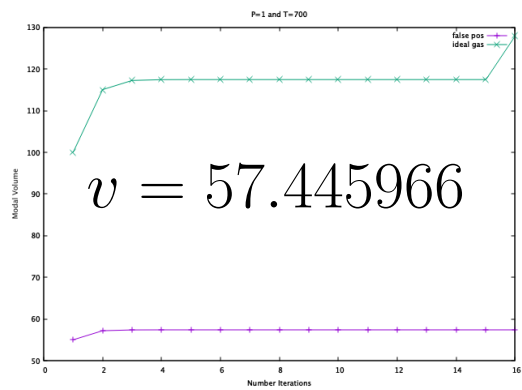


Figure 5: $P = 1atm$, $T = 700K$

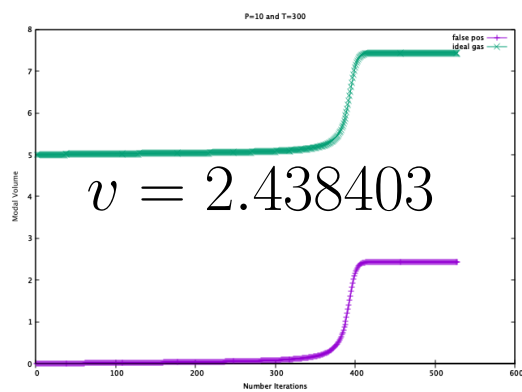


Figure 6: $P = 10atm$, $T = 300K$

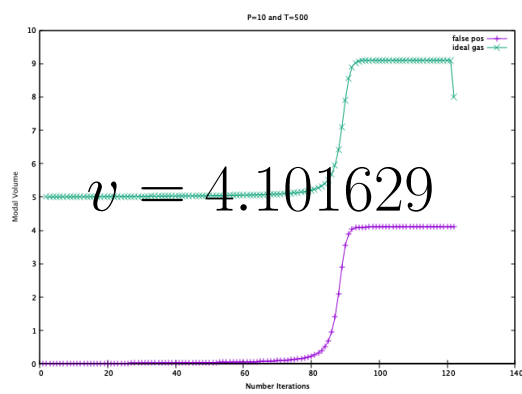


Figure 7: $P = 10atm$, $T = 500K$

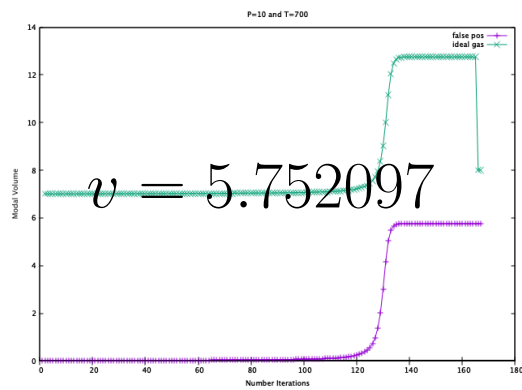


Figure 8: $P = 10atm$, $T = 700K$

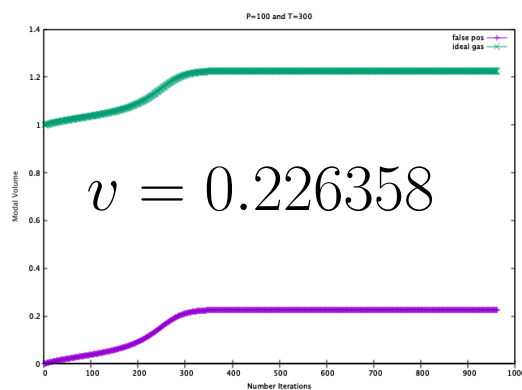


Figure 9: $P = 100atm$, $T = 300K$

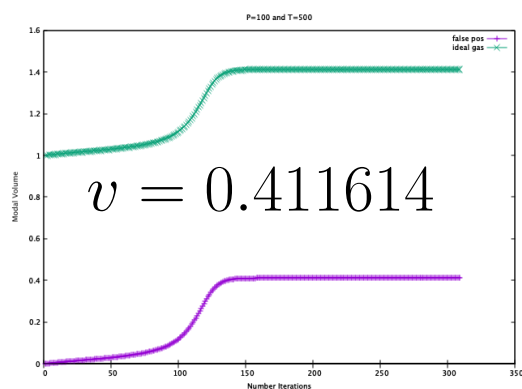


Figure 10: $P = 100atm$, $T = 500K$

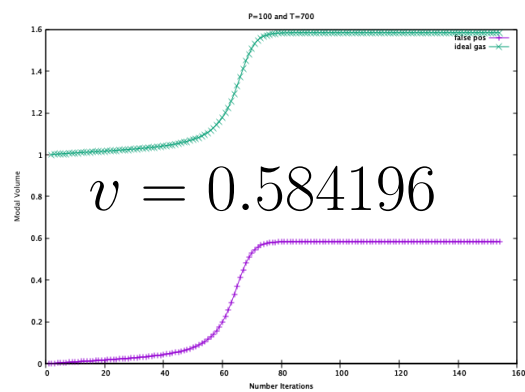


Figure 11: $P = 100atm$, $T = 700K$

Appendix C

```
1  program waals_equation
2      implicit none
3      doubleprecision R, a, b, true
4      integer n, P, T, count
5      doubleprecision x_o, x_n, tol, err, r1, r2
6      doubleprecision step, func
7      doubleprecision x_i, diff
8
9      !INITIALIZE VARIABLES
10     R = 0.082054      !atm/(mol*K)
11     a = 1.360
12     b = 0.03183
13     P = 10            !atm
14     T = 300           !K
15
16     ! Do not know if this is that actual true value. This is just as exact as
17     !!!!! Fortran can get
18     true = 2.4384037628086563
19
20     !fortran can print out up to 16 decimal places so this will produce a
21     !!!!!value as exact as possible with fortran
22     tol = 10.0**(-16)  !tolerance
23     r1 = 0.0           !left solution range value
24     r2 = 1.0           !right solution range value
25     x_n = r1 + r2      !initial estimate
26
27     count = 0
28
29
30     !record basic function values from v = [2.0, 2.5]
31     ! do n = 1, 200
32     ! !function:
33     ! !waals(x_n, R, a, b, P, T)
34     !
35     ! !plot f(v)
36     ! step = 0.5 + (n * 0.01)
37     ! func = waals(step, R, a, b, P, T)
38     ! write(*, *) n, step, func
39     ! write(7, *) step, func
40     ! end do
41
42     !possible P and T values
43     !P = 1, 10, 100
44     !T = 300, 500, 700
45     P = 100; T = 700
46     !estimate the roots, maximum of 500 iterations
47     estimate : do n = 1, 1000
48         !function:
49         !waals(x_n, R, a, b, P, T)
50
51         x_o = x_n
```

```

46      !estimate the roots, maximum of 500 iterations
47      estimate : do n = 1, 1000
48          !function:
49          !waals(x_n, R, a, b, P, T)
50
51          x_o = x_n
52
53          !!!bisection method
54          !x_n = (r1 + r2) / 2
55
56          !!!false position method
57          x_n = (waals(r2, R, a, b, P, T) * r1) - (waals(r1, R, a, b, P, T) * r2)
58          x_n = x_n / (waals(r2, R, a, b, P, T) - waals(r1, R, a, b, P, T))
59          !false position with Ideal Gas Law
60          x_i = (ideal(r2, R, P, T) * r1) - (ideal(r1, R, P, T) * r2)
61          x_i = x_i / (ideal(r2, R, P, T) - ideal(r1, R, P, T))
62
63          !calculate error and print
64          err = abs(x_o - x_n) !error
65          !write(*, *) n, err, x_n
66
67          !calculate difference and print
68          diff = abs(x_n - x_i)
69          write(*, *) n, diff
70          write(7, *) n, x_n
71          write(8, *) n, x_i
72
73          !!!!!!! Print all the things !!!!!!!
74          !!plot error vs time
75          !write(7, *) n, err !bisection
76          !write(8, *) n, err !false pos
77
78          !determine if error is small "enough"
79          if(err < tol .and. n > 1) then
80              write(*, *) "What I found: ", x_n
81              !do not know the true value
82              !write(*, *) "Actual value: ", true
83              write(*, *) n, " iterations"
84              exit
85          end if
86
87          if(waals(r1, R, a, b, P, T) * waals(x_n, R, a, b, P, T) < 0) then
88              r2 = x_n
89          end if
90          if(waals(r1, R, a, b, P, T) * waals(x_n, R, a, b, P, T) > 0) then
91              r1 = x_n
92          end if
93      end do estimate
94
95      CALL SYSTEM('gnuplot script.sh')
96

```

```

65      !write(*,*) n, err, x_n
66
67      !calculate difference and print
68      diff = abs(x_n - x_i)
69      write(*,*) n, diff
70      write(7,*) n, x_n
71      write(8,*) n, x_i
72
73      !!!!!!! Print all the things !!!!!!!
74      !!plot error vs time
75      !write(7,*) n, err !bisection
76      !write(8,*) n, err !false pos
77
78      !determine if error is small "enough"
79      if(err < tol .and. n > 1) then
80          write(*,*) "What I found: ", x_n
81          !do not know the true value
82          !write(*,*) "Actual value: ", true
83          write(*,*) n, " iterations"
84          exit
85      end if
86
87      if(waals(r1, R, a, b, P, T) * waals(x_n, R, a, b, P, T) < 0) then
88          r2 = x_n
89      end if
90      if(waals(r1, R, a, b, P, T) * waals(x_n, R, a, b, P, T) > 0) then
91          r1 = x_n
92      end if
93  end do estimate
94
95  CALL SYSTEM('gnuplot script.sh')
96
97  contains
98
99      !van der Waal's equation of state
100     double precision FUNCTION waals(V, R, a, b, P, T)
101     IMPLICIT NONE
102     doubleprecision :: R, a, b, V
103     integer :: P, T
104     waals = P * V**3 - (b * P + R * T) * V**2 + a * V - a * b
105  END FUNCTION waals
106
107     !Ideal Gas Law
108     double precision FUNCTION ideal(V, R, P, T)
109     IMPLICIT NONE
110     doubleprecision :: R, V
111     integer :: P, T
112     ideal = (R * T) / V
113  END FUNCTION ideal
114
115 end program waals_equation

```

```
1  #!/usr/bin/gnuplot
2
3  set title "P=100 and T=700"
4
5  set xzeroaxis linestyle 8 linewidth 2
6  set xtics axis #0.1
7  set xlabel "Number Iterations"
8  set ylabel "Modal Volume"
9  #plot "fort.7" title "f(v)" with linespoints
10 #plot "fort.7" every ::1 title "bisection" with linespoints, "fort.8" every ::1 title "false pos" with linespoints
11 plot "fort.7" title "false pos" with linespoints, "fort.8" title "ideal gas" with linespoints
12
13 #set table "plot.tex"
14 #plot "fort.7"
15 #unset table
16
17 pause -1
18
```