J. Technology and Education, Vol.21, No.2, pp.81–88 (2014)

General Paper

Technical Computing Systems in Chemistry

Michael P. McCann

Coastal Carolina Community College (Jacksonville, NC 28546-6816, USA)

mccannmp@gmail.com

(Received November 2, 2014; Accepted December 1, 2014)

Abstract

Some freely available computer algebra systems (CAS) and technical computing software is explored which may be of use to chemists.

Keywords: upper-division undergraduate, graduate education / research, physical chemistry, calculator-based learning, computer-based learning, computational chemistry, gases, kinetic-molecular theory, mathematics / symbolic mathematics, physical properties, theoretical chemistry

The use of computer algebra systems, CAS, and chemistry has been addressed. [1,2] It was tempting to title this paper, “Variations on a Theme by Noggle and Wood.” [3] The authors demonstrated an elegant use of Mathematica [4] to solve an equation of state for vapor pressure and boiling point. For the occasional user, Mathematica is an expensive piece of software.

There are a number of free alternatives. The oldest of all computer algebra systems is Maxima. [5] It started out in the late 1960's as Macsyma at the Massachusetts Institute of Technology. Maxima is freely available under the General Public License, GPL. [6] There are some versions of Maxima with a nice graphical user interface, GUI, such as wxMaxima. [7] Other free CAS include Sage, [8] Axiom, [9] and Mathomatic [10] to mention a few.

Though not a CAS, Octave [11] was developed by a chemical engineer and uses matrices to do numerical simulations. Julia [12] is a language on a virtual machine for technical computing. Finally, the latest calculators come with a CAS.

Maxima, Octave, Julia and the CAS on an HP-50g calculator were used with the Peng Robinson equation [13] to calculate the vapor pressure and normal boiling point of nitrogen gas.

$$p = \frac{RT}{V_m - b} - \frac{a\alpha}{V_m^2 + 2bV_m - b^2}$$

where

$$R = 0.082056$$

$$a = \frac{0.457235R^2T_c^2}{p_c}$$

$$b = \frac{0.077796RT_c}{p_c}$$

$$\alpha = \left(1 + \kappa\left(1 - \sqrt{T/T_c}\right)\right)^2.$$

For nitrogen

$$T_c = 126.15\text{K}$$

$$p_c = 33.5\text{atm}$$

$$\kappa = 0.44.$$

The Gibbs free energy is minimized using [14]

$$p(V_{gas} - V_{liquid}) - \int p\,(V,T)dV = 0$$

Since the Peng Robinson equation is cubic with regard to volume, when solving the equation for volume, three solutions can be obtained. The smallest volume is that of the liquid, while the largest volume is that of the gas.

The Maxima version (PengRob.wxm) is quite similar to the Mathematica version of Noggle and Wood. [15] There is one significant difference. Maxima uses Newton's method to find roots of an equation. This involves finding the derivative of the original equation. The Maxima command “solve” was never able to find the roots. A routine was added which used the secant method [16] to find roots. This worked quite well but was not quite as simple as the Mathematica [17] command “FindRoot”.

Box 1

Maxima script

```
/*
Joseph H. Noggle and Robert H. Wood, "Calculation of Vapor Pressure Using
 Mathematica," Journal of Chemical Education, Vol. 69, No. 10 (October 1992) pgs. 810-811.

Ding-Yu Peng and Donald B. Robinson, "A New Two-Constant Equation of State,"
 Ind. Eng. Chem. Fundam., Vol. 15, No. 1 (1976) pgs. 59-64.
*/
Time1:elapsed_real_time()$
load(newton1)$
/*
Gas Constant
*/
R:0.0820574$
accuracy:1e-4$
MaxIt:100$
ratprint:false$
/*
Temperature
*/
T:120.0$
Vmin:0.05$
Vmax:0.35$
/*
Nitrogen
*/
Tc:126.19$
Pc:33.53367$
omega: 0.040$
kappa: 0.37464 + 1.54226*omega - 0.26992*omega^2$
b:0.07780*R*Tc/Pc$
chop(ex):=
if numberp(ex) and abs(ex) < 5.0e-15 then 0
elseif mapatom(ex) then ex
else map(chop,ex)$
alpha(T):=(1+kappa*(1-sqrt(T/Tc)))^2$
a(T):=(0.45724*(R^2)*(Tc^2)/Pc)*alpha(T)$
pf(vm,T):=R*T/(vm-b)-a(T)/(vm*(vm+b)+b*(vm-b))$
wxplot2d( pf(vm,T), [vm, Vmin, Vmax]);
vre(p,T):= sort( float( map( rhs,realpart( solve(
[pf(vm,T)-p = 0], vm)))))$
vf(p,T):=[lmax(vre(p,T)),lmin(vre(p,T))]$
fmin(p,T):=(vg:first(vf(p,T)), vl:second(vf(p,T)),
  chop(float(p*(vg-vl)- first(quad_qags
(pf(v,T), v, vl, vg)))))$
p1: 23.0$
p2: 26.0$
if fmin(p1,T) < fmin(p2,T) then ( xlower:p1,
 xupper:p2 ) else ( xlower:p2, xupper:p1 )$
VaporPressure: xupper$
i:1$
while ( (abs( ( xupper - xlower ) / 2)
 > accuracy or notequal(fmin(VaporPressure,T),0)) and i <=
MaxIt)
do  ( VaporPressure: xupper - ( xupper - xlower)
 * fmin(xupper,T) / ( fmin(xupper,T) - fmin(xlower,T)),
 xlower: xupper, xupper: VaporPressure, i:i+1 )$
fpprintprec:4$
VaporPressure;
p:1.0$
T1:70.0$
T2:80.0$
if fmin(p,T1) < fmin(p,T2) then ( xlower:T1, xupper:T2 )
 else ( xlower:T2, xupper:T1 )$
BoilingPoint: xupper$
i:1$
while ( (abs( ( xupper - xlower ) / 2)
 > accuracy or notequal(fmin(p,BoilingPoint),0)) and i <=
MaxIt)
do
  ( BoilingPoint: xupper - ( xupper - xlower)
 * fmin(p,xupper) / ( fmin(p,xupper) - fmin(p,xlower) ),
 xlower: xupper, xupper: BoilingPoint, i:i+1 )$
BoilingPoint;
Time2:elapsed_real_time()$
(Time2 - Time1) / 60;
```

The Octave version (PengRob.m) was similar to the Maxima version and it too needed an added routine employing the secant method to find the roots. You have to be careful defining variables as the letter “a” could represent a scalar or a matrix. The Julia version (PengRob.jl) worked well also.

Box 2

Octave script

```
clear -all
global time0;
global ElapsedTime;
time0 = clock ();
global R=0.082056;
global T=120.0;
global Vmin=0.05;
global Vmax=0.35;
global Tc=126.19;
global Pc=33.53367;
global omega=0.040;
global kappa=0.37464 + 1.54226 * omega - 0.26992 *
omega^2;
global a=0.457235*(R^2)*(Tc^2)/Pc;
global b=0.07780*R*Tc/Pc;
global p=1.0;
global acc=1e-6;
global loops=100;

function retval = alpha(T)
  global kappa Tc
  retval =(1+kappa*(1-sqrt(T/Tc)))^2;
endfunction
function retval = pf(vm)
    global a b R T
    retval = R * T ./ (vm .- b) .- a * alpha(T) ./ ( vm .*
(b .+ vm) .+ b * (-b .+ vm));
endfunction
x=linspace(Vmin,Vmax,500);
T=120.0;
plot(x, pf(x))
function retval = gfe(pm,Tm)
  global a b R T
  T = Tm;
  prp = [pm, pm*b-R*Tm, a*alpha(Tm) - 3*pm*b*b -
2*b*R*Tm, b*b*b*pm + b*b*R*Tm - a*b*alpha(Tm)];
  volpr = sort(roots(prp));
  volliq = volpr(1);
  volgas = volpr(3);
  [intgrl,ierror,nfneval] = quad ("pf", volliq, volgas);
  retval = pm*(volgas-volliq) - intgrl;
endfunction
p1=23.0;
p2=26.0;
T=120.0;
if (gfe(p1,T) < gfe(p2,T))
  xlower=p1;
  xupper=p2;
else
  xlower=p2;
  xupper=p1;
endif
NewPressure=xupper;
i=1;
while ((abs(xupper-xlower)/2 > acc) && (i<=loops) &&
(gfe(NewPressure,T) != 0))
  NewPressure = xupper - ( xupper - xlower ) *
gfe(xupper,T) / ( gfe(xupper,T) -gfe(xlower,T));
  xlower = xupper;
  xupper = NewPressure;
  VaporPressure = NewPressure;
  i++;
endwhile
VaporPressure
p=1.0;
T1=70.0;
T2=80.0;
if (gfe(p,T1) < gfe(p,T2)) quit
  xlower=T1;
  xupper=T2;
else
  xlower=T2;
  xupper=T1;
endif
NewTemp=xupper;
i=1;
while ((abs(xupper-xlower)/2 > acc) && (i<=loops) &&
(gfe(p,NewTemp) !=0))
  NewTemp = xupper - ( xupper - xlower ) * gfe(p,xupper)
/ ( gfe(p,xupper) -gfe(p,xlower));
  xlower = xupper;
  xupper = NewTemp;
  BoilingPoint = NewTemp;
  i++;
endwhile
BoilingPoint
ElapsedTime = etime (clock (), time0);
ElapsedTime
```

Box 3

Julia script

```
# Julia program, "PengRob.jl"

tic()
using Roots
using Polynomial
using Winston
global VaporPressure
global BoilingPoint
const R = 0.082056
const T = 120.0
const Vmin = 0.05
const Vmax = 0.35
const Tc = 126.19
const Pc = 33.53367
const omega = 0.040
const kappa = 0.37464 + 1.54226 * omega - 0.26992 *
omega^2
```

```
const a = 0.457235*(R^2)*(Tc^2)/Pc
const b = 0.07780*R*Tc/Pc
const p = 1.0
const acc = 1e-6
const loops = 1000

function alpha(T)
  (1+kappa*(1-sqrt(T/Tc)))^2
end
function pf(Vol,Tptr)
  R*T/(Vol-b)-a*alpha(Tptr)/(Vol*(b+Vol)+b*(Vol-b))
end
x=linspace(Vmin,Vmax,500)
n = size(x, 1)
y = copy(x)
for i=1:n
  y[i]=pf(x[i],T)
end
plot(x,y)
function gfe(pm,Tm)
  fpr = Poly([pm, (pm*b-R*Tm), (a*alpha(Tm) - 3*pm*b*b -
2*b*R*Tm), (b*b*b*pm + b*b*R*Tm - a*b*alpha(Tm))])
  z,l = fzero(fpr)
  volliq = z[1]
  volgas = z[3]
  function lpf(x)
    R*Tm/(x-b)-a*alpha(Tm)/(x*(b+x)+b*(x-b))
  end
  intgrl = quadgk (lpf, volliq, volgas)
  return pm*(volgas-volliq) - intgrl[1]
end
p1=23.0
p2=26.0
if (gfe(p1,T) < gfe(p2,T))
  xlower=p1
  xupper=p2
else
  xlower=p2
  xupper=p1
end
NewPressure=xupper
i=1
while ((abs(xupper-xlower)/2 > acc) && (i<=loops) &&
(gfe(NewPressure,T) != 0))
  NewPressure = xupper - ( xupper - xlower ) * gfe(xupper,T) /
( gfe(xupper,T) -gfe(xlower,T))
  xlower = xupper
  xupper = NewPressure
  VaporPressure = NewPressure
  i += 1
end
@printf("Vapor pressure = %2.2f atm", VaporPressure)
print("\n\r")
T1=70.0
T2=80.0
if (gfe(p,T1) < gfe(p,T2))
  xlower=T1
  xupper=T2
else
  xlower=T2
  xupper=T1
end
NewTemp=xupper
i=1
while ((abs(xupper - xlower) / 2 > acc) && (i<=loops) &&
(gfe(p,NewTemp) !=0))
  NewTemp = xupper - ( xupper - xlower ) * gfe(p,xupper) /
( gfe(p,xupper) - gfe(p,xlower))
  xlower = xupper
  xupper = NewTemp
  BoilingPoint = NewTemp
  i += 1
end
@printf("Boiling point = %2.2f K", BoilingPoint)
print("\n\r")
toc()
```

Writing a program using the CAS on a calculator was the most difficult. Fortunately on the HP 50g calculator there is a debugger [18] that can be used to write programs and also an emulator [19] where the program can be run before transferring it to the calculator. Writing the program on the calculator using only the keys on the calculator and the LCD screen would have been beyond one's patience. The calculator version (PengRob.s) was the least elegant. The CAS on the calculator did have some functions that saved some steps such as "∫" for integration rather than having to write a routine to do the integration.

Only the script in Box 4 works on the HP calculator. The other scripts, Boxes 1-3, the Maxima, Octave and Julia scripts work on a computer and do not work on the HP calculator.

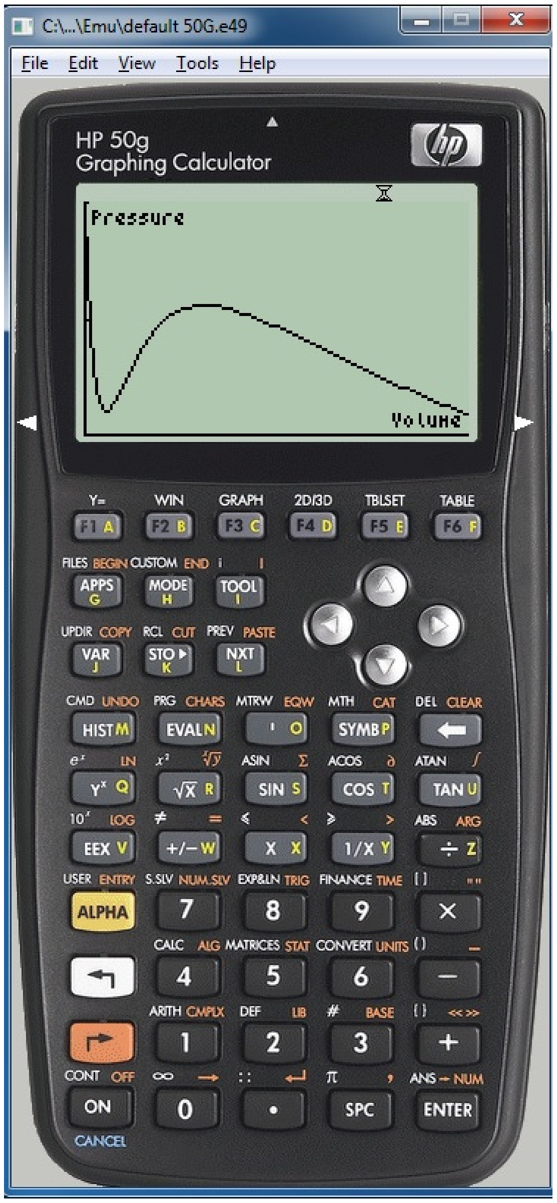
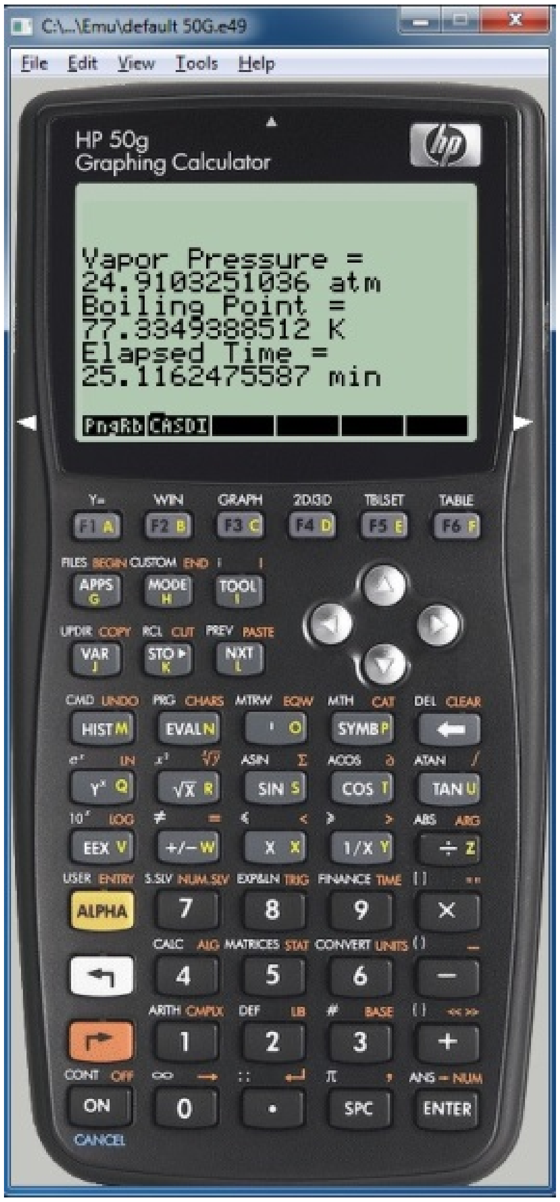

C:\...\Emu\default 50G.e49
File Edit View Tools Help
HP 50g Graphing Calculator
Pressure
Volume

Fig.1


C:\...\Emu\default 50G.e49
File Edit View Tools Help
HP 50g Graphing Calculator
PngRb CASDI

```
Vapor Pressure =
24.9103251036 atm
Boiling Point =
77.3349388512 K
Elapsed Time =
25.1162475587 min
```

Fig.2

Box 4
HP RPL program

```
%%HP: T(0)A(D)F(.);
@ You may edit the T(0)A(D)F(.) parts.
@ The earlier parts of the line are used by Debug4x.
@
@ This program uses an equation of state for a gas.  In this case,
@   it is using the Peng-Robison equation.  This equation is used
@   to calculate the vapor pressure at a particular temperature
@   and it also calculates the normal boiling point.
@
@ The constants used here are for nitrogen gas (N2)
@
@ This is based on the following paper:
@   "Calculation of Vapor Pressure Using Mathematica" by Joseph H.
@   Noggle and Robert H. Wood, Journal of Chemical Education, Vol.
@   69, No. 10, October 1992, pgs. 810-811.
```

```
@
@ This program is written in User RPL for an HP50g calculator
@   in RPN mode.  The calculator should not be in exact mode.
@   MODE  CAS  Approx (check)  If you forget to do this, the
@   program will stop and ask to switch to approximate mode.
@
@ Michael P. McCann
@ 22 March 2013
<<
        { R T P X Vmin Vmax Tc Pc K ALPHA A B PF PvsV GasPlot EQ } PURGE
        { PPAR p1 p2 VRE VL VG FMIN IERR ACR XLOW XUP VapP BoilT } PURGE
        { FM1 FM2 T1 T2 TM1 } PURGE
        TICKS TM1 STO                  @ Start timing
        1E-6 ACR STO                   @ Desired accuracy
        23 P1 STO                      @ Lower pressure limit
        26 P2 STO                      @ Upper pressure limit
        70 T1 STO                      @ Lower temperature limit
        80 T2 STO                      @ Upper temperature limit
        P1 XLOW STO                    @ Variable used to find the root
        P2 XUP STO                     @ Variable used to find the root
        P1 VapP STO                    @ Vapor pressure
        120 T STO                      @ Initial temperature
        T BoilT STO                    @ Boiling temperature
        0.0820574 R STO                @ Ideal Gas Constant
        1. P STO                       @ pressure
        1. VL STO                      @ volume of the liquid
        1. VG STO                      @ volume of the gas
        .05 Vmin STO                   @ minimum volume
        .35 Vmax STO                   @ maximum volume
        126.19 Tc STO                  @ critical temperature
        33.53367 Pc STO                @ critical pressure
        .04 → ω                        @ omega (local variable)

        <<                             @ function to calculate kappa
          ω SQ .26992 * NEG
          ω 1.54226 * +
         .37464 +
        >> K STO
                                       @ function to calculate 'b'
         .07780 R * Tc * Pc /
        >> B STO
        << → t                         @ function to calculate 'alpha(T)'
            <<                         @ temperature should be on the stack
             t Tc /  √ NEG 1 +         @ 't' is local variable for temperature
             K * 1 + SQ
            >>
        >> 'ALPHA' STO

        << → t                         @ function to calculate 'a(T)'
            <<                         @ temperature should be on the stack
            t ALPHA 0.45724 *           @ 't' is local variable for temperature
            R SQ * Tc SQ *             @ put 't' on stack, call 'alpha'
            Pc /
            >>
        >> 'A' STO

        <<                             @ function to calculate pressure 'pf(V,T)'
          R T * X B - /                @ 'X' is volume
          T A X B + X *
          X B - B * + / -
        >> 'PF' STO
        <<                             @ function to plot P vs. V
         PF STEQ
         FUNCTION Vmin Vmax XRNG
         18 34 YRNG
```

```
 ERASE (0.05,18)
                     {(0.05,18) 8 "Volume" "Pressure"} AXES
                     LABEL DRAX DRAW
 PICT RCL PvsV STO
>> GasPlot STO
GasPlot
CLEAR

<< → ta pa                              @ temperature and pressure should be
   <<                                   @ be put on the stack before calling
                      R ta * X B - /    @ this function
                      ta A X B + X *
                      X B - B * + / -   @ function to find the solutions,
                      pa - X ZEROS      @ or volume, where PF-p=0
                      DUP SORT          @ a list is returned
                      HEAD 'VL' STO     @ duplicate the list and sort
                      REVLIST           @ VL is the volume of the liquid
                      HEAD 'VG' STO     @ reverse sort copy the list
                      VG VL - pa * VL VG R ta * X B - / ta ALPHA 0.45724 *
                      R SQ * Tc SQ *
                      Pc / X B + X * X B - B * + / - X ò -
    >>                                  @ VG is the volume of the gas
>> FMIN STO
P1 'XLOW' STO
P2 'XUP' STO
T P1 FMIN 'FM1' STO
T P2 FMIN 'FM2' STO
IF FM1 FM2 <
THEN P1 'XLOW' STO ; P2 'XUP' STO
ELSE P1 'XUP' STO ; P2 'XLOW' STO
END
XUP 'VapP' STO                    @ find the vapor pressure at 120 K
CLEAR
DO XUP XUP XLOW - T XUP FMIN
 * T XUP FMIN T XLOW FMIN -
                     / - 'VapP' STO
 XUP 'XLOW' STO ; VapP 'XUP' STO
UNTIL XUP XLOW - 2 / ABS ACR <
END
1 'P' STO
T1 'XLOW' STO
T2 'XUP' STO
T1 P FMIN 'FM1' STO
T2 P FMIN 'FM2' STO
IF FM1 FM2 <
THEN T1 'XLOW' STO ; T2 'XUP' STO
ELSE T1 'XUP' STO ; T2 'XLOW' STO
END
XUP 'BoilT' STO                  @ Find the normal boiling point at 1 atm
CLEAR
DO XUP XUP XLOW - XUP P FMIN
 * XUP P FMIN XLOW P FMIN -
                     / - 'BoilT' STO
 XUP 'XLOW' STO ; BoilT 'XUP' STO
UNTIL XUP XLOW - 2 / ABS ACR <
END                              @ Display the results
440 .5 BEEP ; TEXT ; CLEAR
"Vapor Pressure = " CLLCD 3 DISP
VapP STR " atm" + 4 DISP
"Boiling Point = " 5 DISP
BoilT STR " K" + 6 DISP
"Elapsed Time = " 7 DISP
TICKS TM1 - BR 8192 / 60 / STR " min" +
8 DISP 0 WAIT
CLEAR                            @Remove all the variables
```

```
          { R T P X Vmin Vmax Tc Pc K ALPHA A B PF PvsV GasPlot EQ } PURGE
          { PPAR P1 P2 VG VL VRE FMIN IERR ACR VapP BoilT } PURGE
          { FM1 FM2 XUP XLOW T1 T2 TM1 } PURGE
>> HOME PngRb STO                              @ Store as the RPL program "PngRb"
```

Table 1 compares the speed of execution of all four versions.

Table 1

Program version	**Execution time**
Maxima (PengRob.wxm)	8.33 minutes
Octave (PengRob.m)	0.83 minutes
Julia (PengRob.jl)	7.89 seconds
HP 50g (PengRob.s)	38.37 minutes

The Maxima, Julia, and Octave versions were run on a 2.4 GHz PC with a quad core processor (only one core was used) with 8 GB of RAM under Fedora 17 Linux. The HP 50g calculator runs a 75 MHz ARM processor and has 2.5 MB of RAM. The HP 50g calculator does well compared to Maxima on a computer considering the difference in their processor speeds. Neither Octave nor Julia are CAS and thus are not saddled with doing symbolic math so it isn't surprising that they out perform Maxima. Julia seems to live up to its claims of being quite fast.

Each programming language / CAS has its own quirks and it takes some time to get proficient writing a program in that software package. Even with the debugger and emulator on a computer, the calculator version was the most difficult to write. Calculators seem to be a technological dead end. Once an indispensable tool of scientists and engineers, most of their use now is in the classroom. Neither Hewlett Packard nor Texas Instruments has come out with a new calculator in years. The HP 50g was introduced in 2006.

There is some quite capable software and the price is right. If I was teaching a physical chemistry class would I require my students to learn one of these software packages? That is a tough call. I have seen math majors balk at having to learn a CAS. With all the difficulties that students have learning the concepts of physical chemistry, learning one of the above software packages would be an additional burden. A lab or two exploring the capabilities of the above software as applied to physical chemistry might be a nice idea.

References

(1) Hanson, Mervin P. Computer Algebra Systems in Physical Chemistry. The Chemical Educator 1996 1(1) 1-21.

(2) Roussel, Marc R. Redesigning the Quantum Mechanics Curriculum to Incorporate Problem Solving Using a Computer Algebra System. J. Chem. Ed. 1999, 76(10), 1373.

(3) Noggle, Joseph H.; Wood, Robert H. Calculation of Vapor Pressure Using Mathematica, J. Chem. Ed. October 1992 69(10), 810-811.

(4) http://www.wolfram.com/mathematica/

(5) http://maxima.sourceforge.net/

(6) https://www.gnu.org/licenses/gpl.html

(7) http://andrejv.github.io/wxmaxima/

(8) http://www.sagemath.org/

(9) http://www.axiom-developer.org/

(10) http://www.mathomatic.org/

(11) https://www.gnu.org/software/octave/

(12) http://julialang.org/

(13) Peng, Ding-Yu; Robinson, Donald B. A New Two-Constant Equation of State, Ind. Eng. Chem. Fundam. 1976 15(1) 59-64.

(14) Noggle and Wood, op. cit.

(15) Noggle and Wood, op.cit.

(16) Numerical Recipies in C, The Art of Scientific Computing, Press, William H.; Flannery, Brian P.; Teukolsky, Saul A.; Vetterling, William T.; Cambridge University Press (1988) pp 263-266.

(17) mathematica, op.cit.

(18) http://www.debug4x.com/

(19) http://www.hpcalc.org/details.php?id=3644