Solution to PS 13

```
> 
> restart;
> with(plots): with(linalg):
Warning, new definition for norm
Warning, new definition for trace
```

Experimental data

```
> plotdata := (i0,i1,x,y) -> plot([[x['i'],y['i']]
  $'i'=i0..i1],color=black,style=line);
```

$$plotdata := (i0, i1, x, y) \rightarrow \mathrm{plot}([[x_{'i'}, y_{'i'}] \$ ('i' = i0 .. i1)], color = black, style = line)$$

```
> xData := [0.078,0.228,0.303,0.590,0.890];
  yData := [0.062,0.197,0.255,0.531,0.854];
  pData := [39.73,37.07,35.60,32.13,25.45];
  n := coldim([xData]);
```

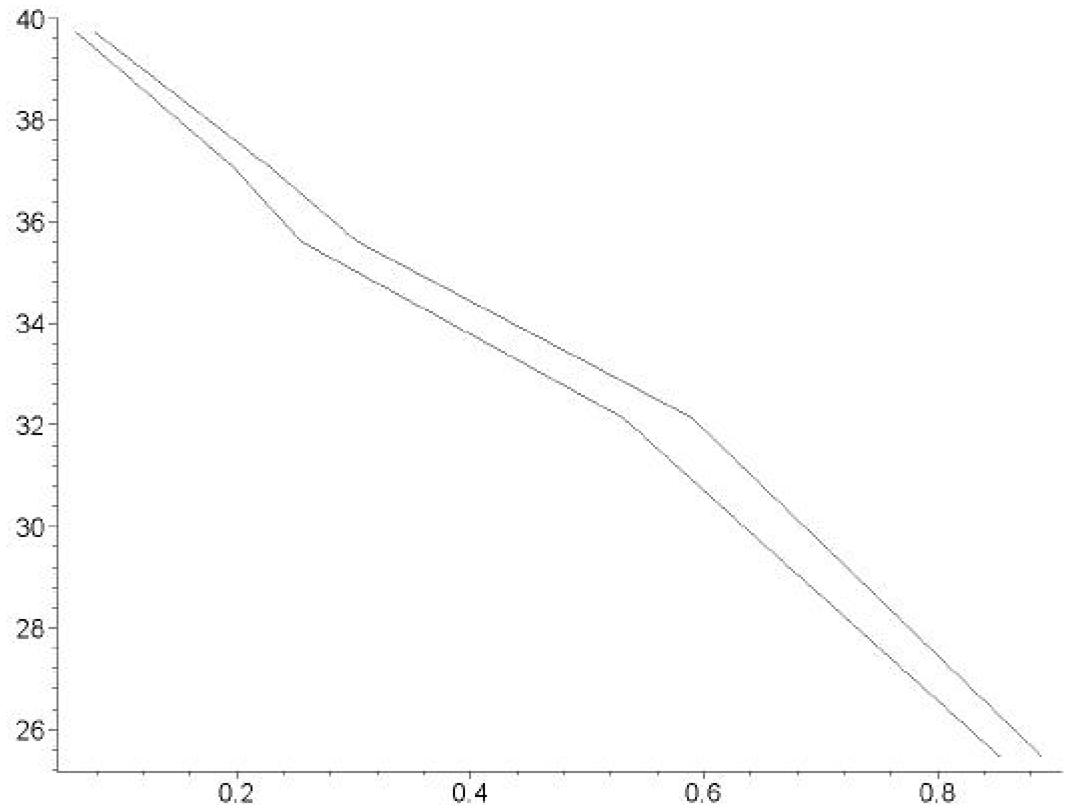
$$xData := [.078, .228, .303, .590, .890]$$

$$yData := [.062, .197, .255, .531, .854]$$

$$pData := [39.73, 37.07, 35.60, 32.13, 25.45]$$

$$n := 5$$

```
> pxData := plotdata(1,n,xData,pData):
  pyData := plotdata(1,n,yData,pData):
> display({pxData,pyData});
```




```
> R := 8.314;
```
J/K-mol (but units don't matter, given the way this calculation is set up!)

$$R := 8.314$$

SRK equation of state

Soave-Redlich-Kwong EOS parameters

```
> epsilon := 0;
  sigma := 1;
  Omegaa := 0.42748;
  Omegab := 0.08664;
  alpha := (Tr,omega) ->
  (1+(0.480+1.574*omega-0.176*omega^2)*(1-sqrt(Tr)))^2;
  A := (T,Tc,Pc,omega) -> Omegaa*alpha(T/Tc,omega)*R^2*Tc^2/Pc;
  B := (Tc,Pc) -> Omegab*R*Tc/Pc;
```

$$\varepsilon := 0$$

$$\sigma := 1$$

$$Omegaa := .42748$$

$$Omegab := .08664$$

$$\alpha := (Tr, \omega) \rightarrow (1 + (.480 + 1.574\,\omega - .176\,\omega^2)\,(1 - \sqrt{Tr}\,))^2$$

$$A := (T, Tc, Pc, \omega) \rightarrow \frac{Omegaa\,\alpha\left(\frac{T}{Tc}, \omega\right) R^2\, Tc^2}{Pc}$$

$$B := (Tc, Pc) \rightarrow \frac{Omegab\ R\ Tc}{Pc}$$

Compressibility factor

```
> Z := (T,V,a,b) -> V/(V-b) - a*V/R/T/(V+epsilon*b)/(V+sigma*b);
```

$$Z := (T, V, a, b) \rightarrow \frac{V}{V-b} - \frac{a\ V}{R\ T\ (V+\varepsilon\ b)\ (V+\sigma\ b)}$$

Liquid and vapor volumes, in same units as the b-parameter

Liquid and vapor volumes, in same units as b

```
> vl := (T,p,a,b) -> min(evalf(solve(Z(T,v,a,b)/v=p/R/T,v)));
  vv := (T,p,a,b) -> max(evalf(solve(Z(T,v,a,b)/v=p/R/T,v)));
```

$$vl := (T, p, a, b) \rightarrow \min\left(\mathrm{evalf}\left(\mathrm{solve}\left(\frac{\mathrm{Z}(T, v, a, b)}{v} = \frac{p}{R\ T}, v\right)\right)\right)$$

$$vv := (T, p, a, b) \rightarrow \max\left(\mathrm{evalf}\left(\mathrm{solve}\left(\frac{\mathrm{Z}(T, v, a, b)}{v} = \frac{p}{R\ T}, v\right)\right)\right)$$

Mixing rules

```
> k12 := 0.00;
  amix := (y,a1,a2) -> y^2*a1 + 2*y*(1-y)*sqrt(a1*a2)*(1-k12) +
  (1-y)^2*a2;
  bmix := (y,b1,b2) -> y*b1 + (1-y)*b2;
  abar1 := (y,a1,a2) -> 2*(y*a1 + (1-y)*sqrt(a1*a2)*(1-k12)) -
  amix(y,a1,a2);
  abar2 := (y,a1,a2) -> 2*(y*sqrt(a1*a2)*(1-k12) + (1-y)*a2) -
  amix(y,a1,a2);
  bbar1 := (y,b1,b2) -> b1;
  bbar2 := (y,b1,b2) -> b2;
>
```

$$k12 := 0$$

$$amix := (y, a1, a2) \rightarrow y^2\ a1 + 2\ y\ (1-y)\ \sqrt{a1\ a2}\ (1-k12) + (1-y)^2\ a2$$

$$bmix := (y, b1, b2) \rightarrow y\ b1 + (1-y)\ b2$$

$$abar1 := (y, a1, a2) \rightarrow 2\ y\ a1 + 2\ (1-y)\ \sqrt{a1\ a2}\ (1-k12) - \mathrm{amix}(y, a1, a2)$$

$$abar2 := (y, a1, a2) \rightarrow 2\ y\ \sqrt{a1\ a2}\ (1-k12) + 2\ (1-y)\ a2 - \mathrm{amix}(y, a1, a2)$$

$$bbar1 := (y, b1, b2) \rightarrow b1$$

$$bbar2 := (y, b1, b2) \rightarrow b2$$

fugacity coefficient for component in mixture

```
> lnPhi := proc(i,T,V,y,a1,a2,b1,b2)
     local a, b, bbar, abar, z;
     a := amix(y,a1,a2);
```

```
    b := bmix(y,b1,b2);
    if(i=1) then
       bbar := bbar1(y,b1,b2);
       abar := abar1(y,a1,a2);
    else
       bbar := bbar2(y,b1,b2);
       abar := abar2(y,a1,a2);
    fi;
    z := Z(T,V,a,b);
    evalf(bbar/b*(z-1) - ln((V-b)*z/V) +
 a/(b*R*T)/(epsilon-sigma)*(1+abar/a-bbar/b)*ln((V+sigma*b)/(V+e
 psilon*b)));
 end;
```

lnPhi := **proc**(*i*, *T*, *V*, *y*, *a1*, *a2*, *b1*, *b2*)

local *a*, *b*, *bbar*, *abar*, *z*;

$a := \text{amix}(y, a1, a2);$

$b := \text{bmix}(y, b1, b2);$

if $i = 1$ **then** $bbar := \text{bbar1}(y, b1, b2);\ abar := \text{abar1}(y, a1, a2)$

else $bbar := \text{bbar2}(y, b1, b2);\ abar := \text{abar2}(y, a1, a2)$

fi;

$z := \text{Z}(T, V, a, b);$

$\text{evalf}(bbar*(z-1)/b - \ln(((V-b)*z)/V)$
$+ a*(1 + abar/a - bbar/b)*\ln((V+\sigma*b)/(V+\varepsilon*b))/(b*R*T*(\varepsilon-\sigma)))$

end

Bubble pressure routine, written according to Fig. 13.2

```
> bubbleP :=
 proc(T,x,pGuess,yGuess,Tc1,Pc1,omega1,Tc2,Pc2,omega2)
   local phi1L, phi1V, phi2L, phi2V, p, y, VL, VV, a1, a2, b1,
 b2, s, K1, K2, sOld;
   a1 := A(T,Tc1,Pc1,omega1);
   a2 := A(T,Tc2,Pc2,omega2);
   b1 := B(Tc1,Pc1);
   b2 := B(Tc2,Pc2);
   y := yGuess;
   s := 1.1;
   p := pGuess/s;
   while abs(s-1) > 1e-4 do
     p := p*s;
 #    print(p);
     VL := vl(T,p,amix(x,a1,a2),bmix(x,b1,b2));
     VV := vv(T,p,amix(y,a1,a2),bmix(y,b1,b2));
```

```
      phi1L := exp(lnPhi(1,T,VL,x,a1,a2,b1,b2));
      phi2L := exp(lnPhi(2,T,VL,x,a1,a2,b1,b2));
      phi1V := exp(lnPhi(1,T,VV,y,a1,a2,b1,b2));
      phi2V := exp(lnPhi(2,T,VV,y,a1,a2,b1,b2));
      K1 := phi1L/phi1V;
      K2 := phi2L/phi2V;
      sOld := 1e10;
      s := K1*x + K2*(1-x);
      while abs(s-sOld) > 1e-4 do
        sOld := s;
        y := K1*x/s;
        VV := vv(T,p,amix(y,a1,a2),bmix(y,b1,b2));
        phi1V := exp(lnPhi(1,T,VV,y,a1,a2,b1,b2));
        phi2V := exp(lnPhi(2,T,VV,y,a1,a2,b1,b2));
        K1 := phi1L/phi1V;
        K2 := phi2L/phi2V;
        s := K1*x + K2*(1-x);
#         print(s);
      od;
#     print();
    od;
    [y,p];
  end:
```

Ethane(1) / ethylene(2) critical properties, acentric factor

```
> tc1 := 305.3; pc1 := 48.72; om1 := 0.100;
  tc2 := 282.3; pc2 := 50.43; om2 := 0.087;
```

$tc1 := 305.3$

$pc1 := 48.72$

$om1 := .100$

$tc2 := 282.3$

$pc2 := 50.43$

$om2 := .087$

Calculation of bubble point according to EOS

```
> yFit := array(1..n);
  pFit := array(1..n);
  for j from 1 to n do
    yp :=
  bubbleP(273.14,xData[j],pData[j],yData[j],tc1,pc1,om1,tc2,pc2,o
  m2);
    yFit[j] := yp[1]:
    pFit[j] := yp[2]:
```

```
od;
```

$$yFit := \text{array}(1..5, [\ ])$$

$$pFit := \text{array}(1..5, [\ ])$$

$$yp := [.06459326401, 39.77413216]$$

$$yFit_1 := .06459326401$$

$$pFit_1 := 39.77413216$$

$$yp := [.1901630069, 36.92346702]$$

$$yFit_2 := .1901630069$$

$$pFit_2 := 36.92346702$$

$$yp := [.2546975521, 35.55546176]$$

$$yFit_3 := .2546975521$$

$$pFit_3 := 35.55546176$$

$$yp := [.5218082193, 30.63000444]$$

$$yFit_4 := .5218082193$$

$$pFit_4 := 30.63000444$$

$$yp := [.8562591741, 25.86827870]$$

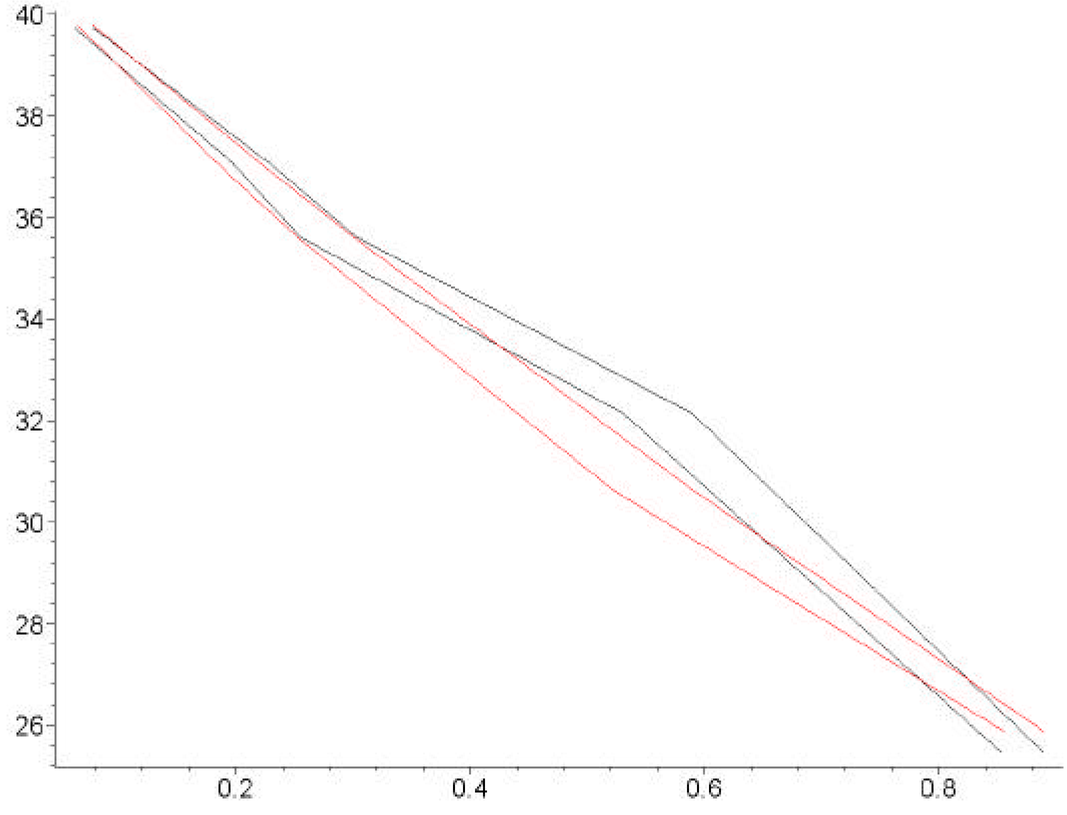
$$yFit_5 := .8562591741$$

$$pFit_5 := 25.86827870$$

Display fitted values (red) with original data (black)

```
> plotdata := (i0,i1,x,y) -> plot([[x['i'],y['i']]
  $'i'=i0..i1],color=red);
  pyFit := plotdata(1,n,yFit,pFit):
  pxFit := plotdata(1,n,xData,pFit):
  display({pxData,pyData,pxFit,pyFit});
```

$$plotdata := (i0, i1, x, y) \rightarrow \text{plot}([[x_{'i'}, y_{'i'}] \$ ('i' = i0 .. i1)], color = red)$$


40
38
36
34
32
30
28
26
0.2
0.4
0.6
0.8