```
> restart;
```
VLE data for chloroform(1)/1,4-dioxane(2) at 50 degC (S,vN&A Table 11.3)

Input data

```
> P :=
  [15.79,17.51,18.15,19.30,19.89,21.37,24.95,29.82,34.80,42.10,60
  .38,65.39,69.36];
```
$P :=$ [15.79, 17.51, 18.15, 19.30, 19.89, 21.37, 24.95, 29.82, 34.80, 42.10, 60.38, 65.39, 69.36]

```
> x1 :=
  [0.0,.0932,.1248,.1757,.2,.2626,.3615,.4750,.5555,.6718,.878,.9
  398,1.0];
```
$x1 :=$ [0, .0932, .1248, .1757, .2, .2626, .3615, .4750, .5555, .6718, .878, .9398, 1.0]

```
> y1 :=
  [0.0,.1794,.2383,.3302,.3691,.4628,.6184,.7552,.8378,.9137,.986
  0,.9945,1.0];
```
$y1 :=$ [0, .1794, .2383, .3302, .3691, .4628, .6184, .7552, .8378, .9137, .9860, .9945, 1.0]

```
> with(linalg): with(plots):with(stats):
Warning, new definition for norm
Warning, new definition for trace
> n := coldim([x1]);
```
$n := 13$

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

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

```
> Pvsx := plotdata(1,n,x1,P):
  Pvsy := plotdata(1,n,y1,P):
  display({Pvsx,Pvsy});
```

Very crude minimization routine for later use

Finds rough minimum of a function of two variables. Examines value of function over an equally-spaced grid of values of the
two variables. "fn" is the function, which should take two arguments; "nPts" is the number of grid points for each variable;
"x1Range" and "x2Range" are lists (of the form [x1min,x1max]) which specify the upper and lower bounds of the grid for each variable.
You can refine the search by running the routine several times, each one with a narrower range of values for the two variables.

```
> crudeMinimize := proc(fn,nPts,x1Range,x2Range)
      local x1,x2,m1,m2,x1step,x2step,obj,bestobj,x1best,x2best;
      x1step := (x1Range[2]-x1Range[1])/nPts;
      x2step := (x2Range[2]-x2Range[1])/nPts;
      bestobj := 1e32;
      for m1 from 0 to nPts do
        x1 := x1Range[1] + m1*x1step;
        for m2 from 0 to nPts do
          x2 := x2Range[1] + m2*x2step;
          obj := fn(x1,x2);
          if(obj < bestobj) then
             bestobj := obj;
```

```
            x1best := x1;
            x2best := x2;
          fi;
        od;
      od;
      evalf([x1best,x2best,bestobj]);
    end;
```

```
crudeMinimize := proc(fn, nPts, x1Range, x2Range)
local x1, x2, m1, m2, x1step, x2step, obj, bestobj, x1best, x2best;
    x1step := (x1Range[2] – x1Range[1]) / nPts;
    x2step := (x2Range[2] – x2Range[1]) / nPts;
    bestobj := .1*10^33;
    for m1 from 0 to nPts do
        x1 := x1Range[1] + m1*x1step;
        for m2 from 0 to nPts do
            x2 := x2Range[1] + m2*x2step;
            obj := fn(x1, x2);
            if obj < bestobj then bestobj := obj; x1best := x1; x2best := x2 fi
        od
    od;
    evalf([x1best, x2best, bestobj])
end
```

Examine and fit Gibbs excess

Determine gE from data

```
> p2sat := 15.79;  p1sat := 69.36;
```

$$p2sat := 15.79$$

$$p1sat := 69.36$$

```
> x2 := [seq(1-x1[i],i=1..n)]:
  y2 := [seq(1-y1[i],i=1..n)]:
  gamma1 := [gam1inf,seq(P[i]*y1[i]/x1[i]/p1sat,i=2..n-1),1]:
  gamma2 := [1,seq(P[i]*y2[i]/x2[i]/p2sat,i=2..n-1),gam2inf]:
  lngamma1 := [seq(ln(gamma1[i]),i=1..n)]:
  lngamma2 := [seq(ln(gamma2[i]),i=1..n)]:
  gE := [seq(x1[i]*lngamma1[i] + x2[i]*lngamma2[i],i=1..n)]:
> gEdataPlot := plotdata(1,n,x1,gE):
> display(gEdataPlot);
```

Fit data to 1-constant and 2-constant (Eq. 11.7) Margules forms

1-constant form

```
gEfit1 := fit[leastsquare[[X,Y], Y=a*X*(1-X)]]([x1,gE]);
```

$$gEfit1 := Y = -.9615256100\, X\,(1 - X)$$

```
> a := -0.9615;
```

$$a := -.9615$$

2-constant form

```
> gEfit2 := fit[leastsquare[[X,Y], Y=(A21*X +
  A12*(1-X))*X*(1-X)]]([x1,gE]);
```

$$gEfit2 := Y = (-.5084474687\ X - .7402510613)\ X\ (1 - X)$$

```
> gamma2vanLaar := x -> exp(A21p/(1 + A21p*(1-x)/A12p/x)^2);
```

$$gamma2vanLaar := x \rightarrow \mathbf{e}^{\left(\frac{A21p}{\left(1+\frac{A21p\,(1-x)}{A12p\,x}\right)^2}\right)}$$

```
> gammavanLaarPlots :=
  plot({gamma1vanLaar(x),gamma2vanLaar(x)},x=0..1):
> display({gamma1dataPlot,gamma2dataPlot,gammavanLaarPlots});
```

Fit to Wilson equation

Formula is not linear in the fitting parameters L12 and L21. Must work with numerical minimization routine (define above).

```
> gEWilson := (x,L12,L21) -> -x*ln(x + (1-x)*L12) -
  (1-x)*ln((1-x) + x*L21);
```

$$gEWilson := (x, L12, L21) \rightarrow -x \ln(x + (1-x)\,L12) - (1-x) \ln(1 - x + x\,L21)$$

```
> gam1inf := gamma1Margules2(0,a12,a21);  gam2inf :=
  gamma2Margules2(1,a12,a21);
```

$$gam1inf := .4769708028$$

$$gam2inf := .2868774953$$

Determine initial guess values for Lambda12 and Lambda21 by matching infinite-dilution activity coefficients

```
> solve({ln(gam1inf)=-ln('L12')+1-'L21',ln(gam2inf)=-ln('L21')+
  1-'L12'},{'L12','L21'});
```

$$\{L12 = .0004390265140,\ L21 = 9.471250750\}$$

```
> l12 := 0.000439; l21 := 9.4713;
```

$$l12 := .000439$$

$$l21 := 9.4713$$

Define functions that give activity coefficients as a function of mole fraction an parameter values

```
> gamma1Wilson := (x,L12,L21) -> exp(-ln(x + (1-x)*L12) +
  (1-x)*(L12/(x + (1-x)*L12) - L21/((1-x) + x*L21)));
  gamma2Wilson := (x,L12,L21) -> exp(-ln((1-x) + x*L21) -
  x*(L12/(x + (1-x)*L12) - L21/((1-x) + x*L21)));
```

$$gamma1Wilson := (x, L12, L21) \rightarrow \mathbf{e}^{\left(-\ln(x+(1-x)\,L12)+(1-x)\left(\frac{L12}{x+(1-x)\,L12}-\frac{L21}{1-x+x\,L21}\right)\right)}$$

$$gamma2Wilson := (x, L12, L21) \rightarrow \mathbf{e}^{\left(-\ln(1-x+x\,L21)-x\left(\frac{L12}{x+(1-x)\,L12}-\frac{L21}{1-x+x\,L21}\right)\right)}$$

Check that functions match infinite-dilution values

```
> gamma1Wilson(0,l12,l21);
```

$$.4769761187$$

```
> gamma2Wilson(1,l12,l21);
```

$$.2868760113$$

Check to see how well these coefficients reproduce the gE and activity-coefficient data

```
> gEWilsonPlot := plot(gEWilson(x,l12,l21),x=0..1):
> display({gEWilsonPlot,gEdataPlot});
> gammaWilsonPlots :=
  plot({gamma1Wilson(x,l12,l21),gamma2Wilson(x,l12,l21)},x=0..0
  .99):
> display({gamma1dataPlot,gamma2dataPlot,gammaWilsonPlots});
```

It indeed gets the infinite-dilution values right, but overall the fit is incredibly bad!
Try again to determine coefficients, now fitting to complete set of activity-coefficient data.

```
> objective := (L12,L21) -> sum(
  evalf(ln(gamma1Wilson(x1['i'],L12,L21)/gamma1['i'])^2 +
  ln(gamma2Wilson(x1['i'],L12,L21)/gamma2['i'])^2),'i'=1..n);
```

$$objective := (L12, L21) \rightarrow \sum_{'i'=1}^{n} \mathrm{evalf}\left(\ln\left(\frac{\mathrm{gamma1Wilson}(x1_{'i'}, L12, L21)}{\gamma 1_{'i'}}\right)^2 + \ln\left(\frac{\mathrm{gamma2Wilson}(x1_{'i'}, L12, L21)}{\gamma 2_{'i'}}\right)^2\right)$$

```
> objective(l12,l21);
```

7.252614302

Minimize the objective function with respect to the lambda's, taking twenty values of each between zero and 20

```
> crudeMinimize(objective,20,[0.0001,20],[0.0001,20]);
```

[2.000090000, 1.000095000, .2421139087]

```
> l12 := 2.4;  l21 := 0.95;
```

$$l12 := 2.4$$

$$l21 := .95$$

Now refine the estimates near the rough minimum

```
> crudeMinimize(objective,10,[1,3],[.5,2]);
```

[2.400000000, .9500000000, .05387353182]

Here are the best-fit values. Let's try them out.

```
> l12 := 2.4;  l21 := 0.95;
```

$$l12 := 2.4$$

$$l21 := .95$$

```
> gEWilsonPlot := plot(gEWilson(x,l12,l21),x=0..1):
> display({gEWilsonPlot,gEdataPlot});
> gammaWilsonPlots :=
  plot({gamma1Wilson(x,l12,l21),gamma2Wilson(x,l12,l21)},x=0..0
  .99):
> display({gamma1dataPlot,gamma2dataPlot,gammaWilsonPlots});
```

Much better!

Consistency test

```
> gam1inf := gamma1fit(0);  gam2inf := gamma2fit(1);
```

$$gam1inf := .4769708028$$

$$gam2inf := .2868774953$$

```
> lnRatio := [seq(ln(gamma1[i]/gamma2[i]),i=1..n)]:
> plotdata(1,n,x1,lnRatio);
> trapezoid := (i0,i1,x,y) ->
  sum(0.5*(y['i'+1]+y['i'])*(x['i'+1]-x['i']),'i'=i0..i1-1);
```

$$trapezoid := (i0, i1, x, y) \rightarrow \sum_{'i'=i0}^{i1-1} (.5\,(y_{'i'+1} + y_{'i'})\,(x_{'i'+1} - x_{'i'}))$$

```
> trapezoid(1,n,x1,lnRatio);
```

-.05366707990

```
> -trapezoid(1,9,x1,lnRatio)+trapezoid(9,n,x1,lnRatio);
```

.5074183619

Here we compare the adifference in the area above and below to the total area above plus below. Should be less than 0.02 or so.

```
> abs("")/";
```

.1057649544

The data don't pass the test!

VLE

Here we examine the 2-parameter Margules model as to how well it re-creates the original P-x-y data. We also examine the prediction of the ideal-solution model.

```
> PfitMargules2 := (x,A12,A21) ->
  p1sat*x*gamma1Margules2(x,A12,A21) +
  p2sat*(1-x)*gamma2Margules2(x,A12,A21);
```

$$PfitMargules2 := (x, A12, A21) \rightarrow p1sat\ x\ \text{gamma1Margules2}(x, A12, A21) + p2sat\ (1 - x)\ \text{gamma2Margules2}(x, A12, A21)$$

```
> yfitMargules2 := (x,A12,A21) ->
  x*gamma1Margules2(x,A12,A21)*p1sat/PfitMargules2(x,A12,A21);
```

$$yfitMargules2 := (x, A12, A21) \rightarrow \frac{x\ \text{gamma1Margules2}(x, A12, A21)\ p1sat}{\text{PfitMargules2}(x, A12, A21)}$$

```
> PvsxFit := plot(PfitMargules2(x,a12,a21),x=0..1):
> PvsyFit :=
  plot([yfitMargules2(x,a12,a21),PfitMargules2(x,a12,a21),x=0..1]
  ):
> display({PvsxFit,Pvsx,PvsyFit,Pvsy});
```

Ideal solution

```
> PIdeal := x -> x*p1sat + (1-x)*p2sat;
  yIdeal := x -> x*p1sat/PIdeal(x);
```

$$PIdeal := x \rightarrow x\ p1sat + (1 - x)\ p2sat$$

$$yIdeal := x \rightarrow \frac{x\ p1sat}{\text{PIdeal}(x)}$$

```
> PvsxIdeal := plot(PIdeal(x),x=0..1):
> PvsyIdeal := plot([yIdeal(x),PIdeal(x),x=0..1]):
> display({PvsxIdeal,Pvsx,PvsyIdeal,Pvsy});
```

"Barker's method" applied to 2-constant Margules formula

Here we try Barker's method (discussed in the text). The basic idea is to fit the activity-coefficient model to the P-x data only.
This involves finding the values of the Margules parameters that minimizes the sum-of-square deviation between the model
pressure and the actual data for the pressure.

```
> objective := (A12,A21) -> sum((PfitMargules2(x1['i'],A12,A21)
  - P['i'])^2,'i'=2..n-1);
```

$$objective := (A12, A21) \rightarrow \sum_{'i'=2}^{n-1} (\text{PfitMargules2}(x1_{'i'}, A12, A21) - P_{'i'})^2$$

```
> print(a12,a21);
```

-.7403, -1.2487

```
> objective(a12,a21);
```

2.864612930

```
> optimum :=
  crudeMinimize(objective,20,[0.5*a12,2*a12],[0.5*a21,2*a21]);
```

$$optimum := [-.7032850000, -1.373570000, .7399279830]$$

```
> a12 := optimum[1]; a21 := optimum[2];
```

$$a12 := -.7032850000$$

$$a21 := -1.373570000$$

```
> optimum :=
  crudeMinimize(objective,20,[0.8*a12,1.2*a12],[0.8*a21,1.2*a21
  ]);
```

optimum := [-.7173507000, -1.373570000, .7204838368]

```
> a12 := optimum[1]; a21 := optimum[2];
```

a12 := -.7173507000

a21 := -1.373570000

Not a dramatic difference, but visible

```
> gammaMargules2Plots :=
  plot({gamma1Margules2(x,a12,a21),gamma2Margules2(x,a12,a21)},
  x=0..1):
> display({gamma1dataPlot,gamma2dataPlot,gammaMargules2Plots});
> PvsxFitBarker :=
  plot(PfitMargules2(x,a12,a21),x=0..1,color=green):
> PvsyFitBarker :=
  plot([yfitMargules2(x,a12,a21),PfitMargules2(x,a12,a21),x=0..
  1],color=green):
> display({PvsxFit,Pvsx,PvsyFit,Pvsy,PvsxFitBarker,PvsyFitBarke
  r});
```