I&EC research
Industrial & Engineering Chemistry Research

Article

pubs.acs.org/IECR

Calculation of Densities from Cubic Equations of State: Revisited

Ulrich K. Deiters*

Institute of Physical Chemistry, University of Cologne, Luxemburger Strasse 116, 50939 Köln, Germany

Ricardo Macías-Salinas

ESIQIE, Departamento de Ingeniería Química, Instituto Politécnico Nacional, Zacatenco, México D. F., 07738, México

Supporting Information

ABSTRACT: Cubic equations of state are still much used in chemical engineering. Inverting such an equation, i.e., calculating molar volumes or densities for given pressure and temperature, is an important and frequently invoked operation. A new algorithm is proposed, which—on modern computers—usually performs faster than other algorithms from the literature.

1. INTRODUCTION

The so-called cubic equations of state are a group of thermal equations of state having the general structure

$$p = \frac{RT}{V_m - b} - \frac{a}{V_m^2 + f_1 V_m + f_2} \quad (1)$$

where V_m denotes the molar volume, p is pressure, and T is temperature. R is the universal gas constant and a and b are substance-dependent parameters; a is usually a function of temperature. Setting $f_1 = f_2 = 0$ turns eq 1 into the van der Waals equation, setting $f_1 = 1$ and $f_2 = 0$ yields the Redlich–Kwong equation,[1] and setting $f_1 = 2$ and $f_2 = -1$ yields the Peng–Robinson equation,[2] but there are also equations of state that let the f_i depend on the substances, e.g., the Trebble–Bishnoi–Salim equation.[3]

All these equations of state have in common that multiplying them with their denominators yields a cubic polynomial:

$$A_3 V_m^3 + A_2 V_m^2 + A_1 V_m + A_0 = 0 \quad (2)$$

with

$$A_0 = -f_2(pb + RT) - ab$$

$$A_1 = pf_2 - f_1(pb + RT) + a$$

$$A_2 = pf_1 - (pb + RT)$$

$$A_3 = p$$

The computation of a molar volume for given temperature and pressure thus amounts to finding the roots of this polynomial—a mathematical operation for which reliable algorithms exist. It is therefore not astonishing that cubic equations of state are frequently used, although they leave much to be desired from the viewpoint of statistical thermodynamics. Another reason for the popularity of some cubic equations of state is the availability of group contribution schemes which allow the prediction of substance parameters from chemical structures; examples are the VTPR method (volume-translated Peng–Robinson[4,5]) and the PPR78 method (predictive Peng–Robinson (version of 1978)[6−8]).

Most algorithms for the calculation of phase equilibria of mixtures require the computation of molar volumes from pressure at each iteration step [but not all, fortunately; see ref 9]. For complex calculations, for instance in the context of enhanced oil recovery and reservoir simulations, large numbers of phase equilibrium calculations have to be performed. Then, of course, it is worthwhile to explore efficient ways for finding the roots of cubic polynomials.

In principle, the roots of cubic polynomials can be determined analytically with the so-called Cardano method.[10] But as this method involves the computation of algebraic and trigonometric functions, there have been several attempts to develop faster, iterative algorithms. Many of these have been listed and compared by Olivera-Fuentes,[11] but there are also more recent attempts.[10,12]

The evolution of computer science over the past decades has not merely increased the clock frequency of microprocessors, but also provided them with multiple arithmetic units (so that some operations can be carried out in parallel), more registers, and a sophisticated cache managment. It therefore seems worthwhile to compare the performances of known cubic root finders again *on modern computers* and, if possible, to develop better ones.

2. ITERATIVE CUBIC ROOT FINDERS

We have to stress that the objective of this section is outlining algorithms which reliably find all real roots of a cubic polynomial with arbitrary real coefficients. Naturally, simplifications might be possible if the coefficients are subject to some restrictions.

Received: November 14, 2013
Revised: December 29, 2013
Accepted: January 10, 2014
Published: January 10, 2014

dx.doi.org/10.1021/ie4038664 | *Ind. Eng. Chem. Res.* 2014, 53, 2529−2536

The iterative solution of cubic equations requires several steps: normalization, scaling, initialization, iteration, and deflation.

2.1. Normalization. The first step of the computation of the real roots of a general cubic equation

$$A_3x^3 + A_2x^2 + A_1x + A_0 = 0 \tag{3}$$

is always its normalization, i.e., the division by the coefficient of the cubic term, which leads to

$$x^3 + a_2x^2 + a_1x + a_0 = 0 \qquad \text{with} \quad a_i = \frac{A_i}{A_3} \tag{4}$$

Evidently, this step can be omitted if the polynomial is already normalized.

2.2. Scaling. Some iterative as well as analytic root finders may eventually become inefficient or even unreliable if the coefficients a_i are either very large or all very close to zero. The problem can be minimized by a scaling, i.e., by working with the equation

$$f(x_{sc}) = x_{sc}^3 + b_2x_{sc}^2 + b_1x_{sc} + b_0 = 0 \tag{5}$$

with

$$x_{sc} = \lambda x \quad \text{and} \quad b_i = a_i\lambda^{3-i}$$

Setting

$$\frac{1}{\lambda} = \max(|a_0|^{1/3}, |a_1|^{1/2}, |a_2|) \tag{6}$$

ensures that the largest coefficient b_i (i = 0, 1, 2) becomes +1 or −1. This choice for λ, however, requires the computation of inverse powers, which is a slow operation on most computers. For practical purposes it is sufficient to extract the exponents of the digital representations [C, C++: standard library function frexp()] of the a_i, divide them by i, and use the largest value to scale the coefficients [C, C++: standard library function ldexp()]. A program code example can be found in section 2 in the Supporting Information.

Evidently, this step can be skipped if the polynomial is guaranteed to have coefficients always in the range [−1; +1].

The sensitivity of the root finder to scale issues depends very much on the choice of initial values. This will be addressed in section 2.3.

2.3. Initial Values. In order to locate the first root of the cubic equation with an iterative procedure, it is necessary to have a good initial guess. Figure 1 shows the shapes that a cubic function can possibly exhibit. There is always an inflection point, and the slope at this point can be positive, zero, or negative. Consequently, the function either is strictly monotonic (cases 3.x), has a saddle point (cases 2.x), or has both a maximum and a minimum (cases 1.x); in the latter case, there can be three real roots (cases 1.3−1.5), one root (1.1, 1.7), or two roots, one of them being a double one (1.2, 1.6). [A double root in a pressure isotherm does not correspond to a stable state. Nevertheless, a reliable algorithm should treat this case correctly, for it might occur during the iterative search for a stable state, or in studies of metastable states and stability limits.]

2.3.1. Cauchy Initialization. According to a theorem of Cauchy [also attributed to E. Rouché, who—after Cauchy—proved this to be a special case of his more general theorem on the roots of complex polynomials], all real roots of an nth degree polynomial must lie within the range [−r; +r] with

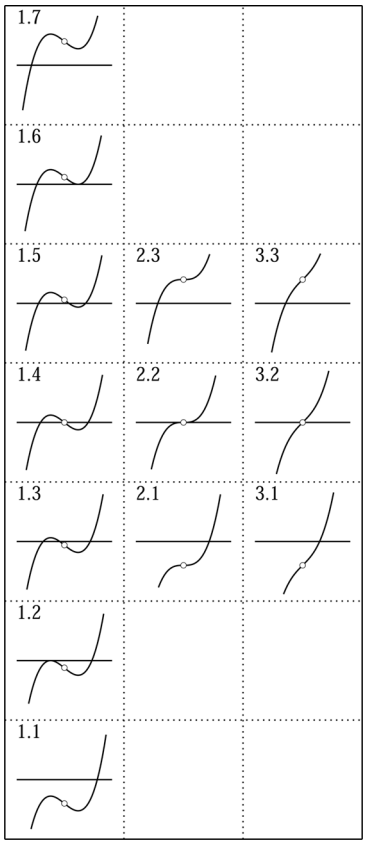

Figure 1. Schematic representation of the various shapes of cubic functions. (○) Inflection point.

$$r = 1 + \max(|a_0|, \ldots, |a_{n-1}|) \tag{7}$$

The inflection point of the cubic polynomial in eq 5 is located at

$$x_{infl} = -\frac{a_2}{3} \tag{8}$$

A reliable way of initializing the iteration is starting from $x = -r$ if $f(x_{infl}) \geq 0$, or from $x = +r$ otherwise. This choice ensures that there are no extrema between the initial guess and the nearest root of the cubic polynomial.[10,12] We shall refer to this initialization method as Cauchy initialization ("C").

2.3.2. Laguerre−Nair−Samuelson Initialization. If there are three real roots, a narrower range is given by the Laguerre−Nair−Samuelson criterion, which for a normalized polynomial is

$$x_{low,high} = -\frac{a_{n-1}}{n} \pm \frac{n-1}{n}\sqrt{a_{n-1}^2 - \frac{2n}{n-1}a_{n-2}} \tag{9}$$

Application to eq 5 results in

$$x_{low,high} = x_{infl} \pm \frac{2}{3}\sqrt{D} \tag{10}$$

with

$$D = b_2^2 - 3b_1$$

But even if x_{low} and x_{high} can be computed ($D > 0$), it is not certain that three roots exist (cases 1.3−1.5 in Figure 1), for cases 1.1 and 1.7 are possible, too. Still, x_{low} or x_{high} can be used as initial guesses.

dx.doi.org/10.1021/ie4038664 | Ind. Eng. Chem. Res. 2014, 53, 2529−2536

If eq 10 has complex solutions only (i.e., if $D < 0$), the cubic polynomial does not have any extrema (cases 3.x). Then, however, it is permissible to use the inflection point as initial value for the iteration. The resulting initialization scheme is therefore

$$x = \begin{cases} x_{\text{low}} & \text{if } D > 0 \ \wedge \ f(x_{\text{infl}}) > 0 \\ x_{\text{infl}} & \text{if } D < 0 \\ x_{\text{high}} & \text{if } D > 0 \ \wedge \ f(x_{\text{infl}}) < 0 \end{cases} \tag{11}$$

Finally, there are some special cases: If $f(x_{\text{infl}}) = 0$, x_{infl} is a root of the polynomial. If $D = 0$, the inflection point has a zero slope (cases 2.x), and then the root is at

$$x = x_{\text{infl}} - \sqrt[3]{f(x_{\text{infl}})} \tag{12}$$

Evidently, no iteration is required in these two cases.

2.3.3. Vieta Initialization. If a normalized cubic polynomial has three real roots, its coefficients can be written according to Vieta's rules:

$$\begin{aligned} a_0 &= -x_1x_2x_3 \\ a_1 &= x_1x_2 + x_2x_3 + x_1x_3 \\ a_2 &= -(x_1 + x_2 + x_3) \end{aligned} \tag{13}$$

If there is ony one real root, x_1, and two complex ones, $x_{2,3} = \alpha \pm i\beta$, the result is similar:

$$\begin{aligned} a_0 &= -x_1(\alpha^2 + \beta^2) \\ a_1 &= \alpha^2 + \beta^2 + 2\alpha x_1 \\ a_2 &= -2\alpha - x_1 \end{aligned} \tag{14}$$

If $|x_1|$ is much smaller than $|x_2|$ and $|x_3|$, the terms containing x_1 can be neglected in the expressions for a_1, and then

$$x_1 \approx -\frac{a_0}{a_1} \tag{15}$$

is a good approximation for the (absolutely) smallest root.

Except for some special applications, such as the calculation of molar volumes from cubic equations of state at very low pressures (see section 3.1), it is usually not known in advance that x_1 is small in comparison with the other roots. The tests for the applicability of this initialization scheme are time-consuming.

2.3.4. "Thermodynamic Initialization". If cubic equations are solved in the context of equations of state, it seems logical to use either zero density (the ideal gas state) or the maximal density (volume equal to equation of state covolume parameter) as initial values. But this is dangerous: The iteration schemes discussed below will converge reliably only if there is no extremum between the initial value and the nearest root. As cubic equations of state, except for the van der Waals equation, can have solutions outside the physically permissible range, starting the iteration at the "thermodynamic bounds" does not necessarily prevent the iteration from running into an extremum and then diverging. [Even the van der Waals equation of state will have nonphysical roots if invoked for negative pressure, which can become necessary when metastable states are studied.]

2.4. Iteration. Once a good initial guess for the first root is available, an iterative scheme can be used to obtain an accurate value. Schemes known to give a good speed of convergence are the following:

1. The Newton–Raphson method, which can be written as follows:

$$x \leftarrow x - \frac{f(x)}{f'(x)} \tag{16}$$

2. Halley's method:

$$x \leftarrow x - \frac{f(x)\,f'(x)}{(f'(x))^2 - \frac{1}{2}f(x)\,f''(x)} \tag{17}$$

3. The "double Newton–Raphson" method by Olivera-Fuentes:[11]

$$x \leftarrow x + \delta_{\text{NR}}\frac{f'(x) + \delta_{\text{NR}}\left(\frac{1}{2}f''(x) + 2\delta_{\text{NR}}\right)}{f'(x) + \delta_{\text{NR}}(f''(x) + 3\delta_{\text{NR}})} \tag{18}$$

where $\delta_{\text{NR}} = -f(x)/f'(x)$ is the correction step of the Newton–Raphson method.

4. Mollerup's method:[13]

$$x \leftarrow x + \delta_{\text{NR}}\left(1 - \frac{f''(x)}{2f'(x)}\delta_{\text{NR}}\right) \tag{19}$$

5. The "modified Richmond" method:[14]

$$x \leftarrow x + \frac{\delta_{\text{NR}}}{1 + \frac{f''(x)}{2f'(x)}\delta_{\text{NR}} + \frac{1}{f'(x)}{\delta_{\text{NR}}}^2} \tag{20}$$

6. The regula falsi method (so-called Illinois variant), also known as the secant method, which needs an additional initial value, x_0; the function values $y = f(x)$ and $y_0 = f(x_0)$ must have opposite signs:

$$x_{\text{new}} = x_0 - \frac{x - x_0}{y - y_0}y_0 \tag{21}$$

with the replacement rules

$$\begin{cases} x_0 \leftarrow x, \quad y_0 \leftarrow y & \text{if } \operatorname{sign}(f(x_{\text{new}})) = \operatorname{sign}(f(x_0)) \\ y_0 \leftarrow \frac{1}{2}y_0 & \text{if } \operatorname{sign}(f(x_{\text{new}})) = \operatorname{sign}(f(x)) \end{cases} \tag{22}$$
$$x \leftarrow x_{\text{new}}, \quad y \leftarrow f(x_{\text{new}})$$

For initialization we use here the inflection point and either $+r$ or $-r$.

7. Steffensen's method (also known as Aitken's delta method),[15] which requires the construction of an auxiliary iteration function, e.g., $g(x) = -(x^3 + a_2x^2 + a_0)/a_1$:

$$\begin{aligned} x_1 &= g(x) \\ x_2 &= g(x_1) \\ x &\leftarrow x - \frac{(x_1 - x)^2}{x_2 - 2x_1 + x_2} \end{aligned} \tag{23}$$

Of course many more iteration methods can be found in the literature. Here we confine ourselves to methods which have at least almost second-order convergence and which can be

dx.doi.org/10.1021/ie4038664 | Ind. Eng. Chem. Res. 2014, 53, 2529–2536

proven to converge reliably if there is no extremum between the initial value and the nearest root.

In this context we have also to mention Muller's method,[16] which is an iterative interpolation scheme like the regula falsi method, but uses parabolic instead of linear interpolation. As it involves solving a quadratic equation at each iteration step, it is slower than the other methods listed above. The Durand–Kerner method,[17] a root finder for polynomials of arbitrary degree, involves complex-number multiplications and divisions at each iteration step; not surprisingly, it turned out to be slower than the other methods by at least 1 order of magnitude.

2.5. Deflation. Once the first real root, x_1, is known, the cubic polynomial can be "deflated", i.e., divided by a linear factor. We set

$$\frac{x^3 + a_2x^2 + a_1x + a_0}{x - x_1} = x^2 + c_1x + c_0 \tag{24}$$

with

$$c_1 = x_1 + a_2 \quad \text{and} \quad c_0 = c_1x_1 + a_1$$

The roots of the resulting quadratic polynomial (if any) can be determined from the well-known analytical formula

$$x_{2,3} = -\frac{c_1}{2} \mp \sqrt{\frac{c_1^2}{4} - c_0} \tag{25}$$

2.6. Remarks. *2.6.1. Evaluation of Polynomials.* In order to save CPU time, Horner's method should be used to compute values of polynomials, e.g.

$$A_3x^3 + A_2x^2 + A_1x + A_0 \rightarrow A_0 + x(A_1 + x(A_2 + xA_3)) \tag{26}$$

Horner's method uses the smallest possible number of multiplications.

2.6.2. Vanishing Third-Order Coefficients. One should avoid the situation where the third-order coefficient, A_3, is small in comparison with the other coefficients, because then the normalizing step will make the lower-order coefficients huge, and that can lead to a loss of precision and even to wrong results.

This can easily happen in the course of vapor pressure calculations close to the triple point, where the pressures are often quite low. If the cubic polynomial is set up according to eq 2, the pressure becomes the third-order coefficient, resulting in an ill-conditioned polynomial which cannot be solved reliably for the roots; even scaling cannot remedy this. Figure 2 illustrates this problem.

Alternatively, the polynomial can be set up in terms of molar densities $\rho = 1/V_m$:

$$A_3 + A_2\rho + A_1\rho^2 + A_0\rho^3 = 0 \tag{27}$$

Now $A_3 = p$ is the zero-order coefficient; if it approaches zero, this will not hamper, but speed up the root finder.

A comparison of ρ-based and V_m-based inversions of the cubic equation has been published before.[10] Incidentally, there are more advantages to a formulation of thermodynamic equations in terms of molar densities (see refs 9 and 18 (Chapter 5.8)).

2.6.3. Very Small Roots. The analytical formula for the roots of the quadratic polynomial, eq 25, obtains them as the sum or the difference of two terms. If one of the roots is very small, it is computed as the difference of two almost equal terms, and may therefore suffer from round-off errors. This problem can be solved by following the computation of the absolutely smaller root with a single Newton iteration step. Alternatively, the problem can be avoided by calculating the smallest root of the cubic polynomial first (iteratively), which can be achieved by using Vieta initialization. For the usual "double precision" arithmetics, this is advisable if $|a_0/a_1| \leq 10^{-5}$.

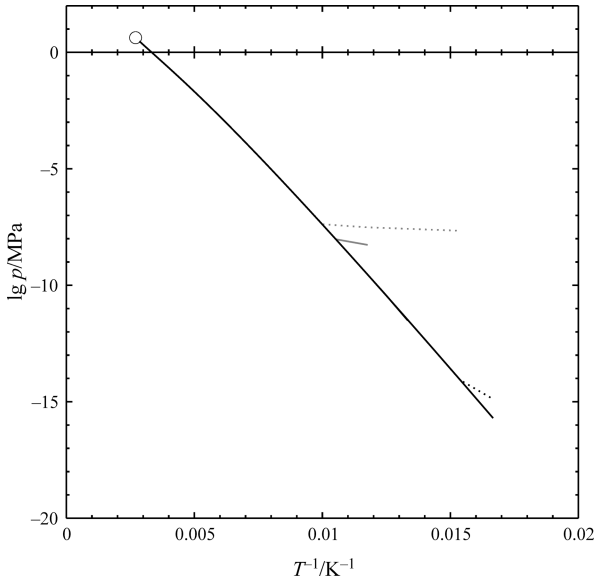

Figure 2. Vapor pressure curve of propane, computed with the Peng–Robinson equation of state.[2] (—) Cardano's method with postiteration (one Newton iteration step), all iterative methods of this work, applied to the density polynomial, eq 27; (···) Cardano's method without postiteration; (gray curves) Cardano's method applied to the volume polynomial, eq 2, with or without postiteration; (○) critical point.

3. RESULTS AND DISCUSSION

3.1. Computing Precision. The most important question is, of course, whether iterative algorithms achieve the same numerical precision as an implementation of the analytical solution (Cardano's method). Figure 2, a log p vs T^{-1} representation of the vapor pressure curve of propane, shows that this is indeed the case: the results for all iterative methods discussed in this work neatly coincide from the critical point down to about 60 K. The only method exhibiting a weakness is the analytical method of Cardano, which shows deviations around 70 K, unless a postiteration (one Newton–Raphson iteration step) is applied.

Cardano's method applied to the volume polynomial, eq 2, leads to grave artifacts in vapor pressure calculations at and below 100 K.

A closer inspection of density calculations at very low pressures, however, shows that all methods encounter difficulties here. Tables 1 and 2 contain liquid and vapor molar volumes of propane, computed with the Peng–Robinson equation of state[2] for pressures along the predicted vapor pressure curve, using "double precision" arithmetics. The calculations were performed with ThermoC software.[19] These results merely serve to demonstrate the numerical problems; we do not claim that the Peng–Robinson equation—with parameters derived from critical data—is accurate at or below triple point conditions.

dx.doi.org/10.1021/ie4038664 | Ind. Eng. Chem. Res. 2014, 53, 2529–2536

Table 1. Calculation of Liquid and Vapor Molar Volumes of Propane at 100 K and 40.6983 mPa (Approximate Vapor Pressure) from the Peng–Robinson Equation of State[2] via the Density Polynomial, eq 27[a]

method	V_m^l/cm^3 mol^{-1}	V_m^g/cm^3 mol^{-1}	p^l/mPa	p^g/mPa
Cardano, postiteration	59.819 34	$2.042\,953 \times 10^{10}$	40.697 87	40.698 30
Newton, LNS/i	59.819 34	$2.042\,128 \times 10^{10}$	40.838 60	40.714 74
Olivera-Fuentes, LNS/i	59.819 34	$2.042\,952 \times 10^{10}$	40.697 87	40.698 32
Halley, LSN/i	59.819 34	$2.042\,952 \times 10^{10}$	40.697 87	40.698 32
Halley, V	59.819 34	$2.042\,953 \times 10^{10}$	40.697 87	40.698 30

[a] p^l and p^g, pressures calculated for these volumes. Initialization schemes: LNS/i, Laguerre–Nair–Samuelson/inflection point; V, Vieta.

Table 2. Calculation of Liquid and Vapor Molar Volumes of Propane at 80 K and 35.6663 μPa (Approximate Vapor Pressure) with the Peng–Robinson Equation of State[2] via the Density Polynomial, eq 27[a]

method	V_m^l/cm^3 mol^{-1}	V_m^g/cm^3 mol^{-1}	p^l/μPa	p^g/μPa
Cardano, postiteration	58.899 61	$1.864\,947 \times 10^{13}$	36.068 84	35.666 30
Newton, LNS/i	58.899 61	$1.790\,272 \times 10^{13}$	53.361 02	37.154 00
Olivera-Fuentes, LNS/i	58.899 61	$1.864\,165 \times 10^{13}$	36.068 84	35.681 28
Halley, LSN/i	58.899 61	$1.864\,144 \times 10^{13}$	36.068 84	35.681 66
Halley, V	58.899 61	$1.864\,947 \times 10^{13}$	36.068 84	35.666 30

[a] See Table 1 for an explanation of the symbols and abbreviations.

Table 3. CPU Times for the Calculation of All Real Roots of a Cubic Polynomial Having Three Real Roots (3r+2x), One Real Root and Two Extrema (1r+2x), or One Real Root and No Extrema (1r+0x) Using Halley's Method: Effects of Scaling and Initialization Scheme[a]

			t_{CPU}/ns		
initialization	scaling	λ	3r+2x	1r+2x	1r+0x
C	off	1	90–136	110	79–95
	off	10^3	154–453	431	367–402
	off	10^{-3}	234	190–207	206–255
C	on		131–180	155	138
LNS/i	off	1	51–62	67	49
	off	10^3	50–67	71	55
	off	10^{-3}	50–67	71	55
LNS/i	on		74–87	94	76
V	n.a.		32–99	55–370	53–103

[a] LNS/i, Laguerre–Nair–Samuelson/inflection point; C, Cauchy; V, Vieta. λ, scaling factor; see section 3 for its definition and a description of the test conditions.

Table 4. CPU Times for the Calculation of All Real Roots of a Cubic Polynomial Having Three Real Roots (3r+2x), One Real Root and Two Extrema (1r+2x), or One Real Root and No Extrema (1r+0x): Comparison of Iteration Methods[a]

		t_{CPU}/ns		
method	C	3r+2x	1r+2x	1r+0x
Cardano		410–680	93	93
Newton–Raphson	2	45–72	86	62
Halley	3	51–64	70	52
Olivera-Fuentes	3	70–96	109	87
Steffensen	2	79–102	111	88
Mollerup	3	65–80	490	92
modified Richmond	3	68–93	113	93
regula falsi	1.6	34–103	245–305	270

[a] Initialization, LNS/i; scaling, off. C, convergence order. See section 3 for a description of the test conditions.

Evidently, the iterative algorithm using Newton's method is not as good at low pressures as the others (or needs more iteration steps in this pressure range). All high-order iterative methods give practically the same results, and they agree well with Cardano's method. It turns out, however, that the pressures calculated from the liquid and vapor molar volumes do not always agree exactly. This phenomenon gets more pronounced for lower pressures. Unfortunately, it cannot be resolved by performing more iteration steps, for it is caused by the structure of the equation of state.

(i) The original cubic equation of state, eq 1, is a difference of two terms. For the liquid states in Tables 1 and 2, these terms amount to 237 or 257 MPa, respectively, while their differences are in the millipascal or even micropascal range and hence very susceptible to rounding errors.

(ii) For the thermodynamic states in Tables 1 and 2, the a_0 coefficient of the normalized cubic polynomial) is of the order of magnitude of $pb/(RT) \approx 10^{-12}$–10^{-8}, whereas the other coefficients are of the order of magnitude 10^{-1}–10^{0}. In this situation, the evaluation of the polynomial involves the addition or subtraction of terms of rather different sizes, and this causes a loss of significant figures.

As a consequence, a molar volume or density that fulfills eq 1 under "double precision" arithmetics does no longer exactly fulfill eq 27, and vice versa. The numerical problems that all algorithms—analytical and iterative—encounter are thus not due to design flaws of these algorithms, but to the structure of cubic equations of state. Working in that pressure range would either require a higher computing precision or a different equation of state.

On the other hand, the iterative methods studied here—after Laguerre–Nair–Samuelson initialization—work reliably for vapor pressure calculations above $0.25T_c$ as well as for density calculations in general. For very low pressures, if $|a_0/a_1| \leq 10^{-6}$, Vieta initialization is advantageous.

3.2. Computing Speed. For the determination of CPU times, sets of coefficients were prepared that resulted in polynoms

- having three real roots (cases 1.2 and 1.4 in Figure 1)
- having one real root and two extrema (cases 1.1 and 1.7)
- or having one real root and no extrema (cases 3.1 and 3.3)

dx.doi.org/10.1021/ie4038664 | Ind. Eng. Chem. Res. 2014, 53, 2529–2536

Data: array of polynomial coefficients, $A_i, i = 0, \ldots 3$
Result: number of real roots n, sorted array of roots $x_i, i = 1, \ldots n$

if $A_3 = 0$ **then**
  $n := \text{SolveQuadratic}(A_0, A_1, A_2; x_1, x_2)$
  return n
end

normalize: $a_i := A_i/A_3, i = 0, 1, 2$
if $a_0 = 0$ **then**
  $x_1 := 0$
  $n := \text{SolveQuadratic}(a_1, a_2, 1; x_2, x_3)$
  if $n > 1$ **then** sort $\vec{x}$
  return $n + 1$
end

calculate inflection point $x_{\text{infl}}, y_{\text{infl}}$
if $y_{\text{infl}} = 0$ **then**
  $x_2 := x_{\text{infl}}$
  $\text{Deflate}(\vec{a}, x_{\text{infl}}; \vec{b})$
  $n := \text{SolveQuadratic}(b_0, b_1, 1; x_1, x_3)$
  return n
end
$D := a_2^2 - 3a_1$
if $D = 0$ **then**
  $x_1 := x_{\text{infl}} - \sqrt[3]{y_{\text{infl}}}$
  return 1
end
$x_1 := x_{\text{infl}}$
if $D > 0$ **then** $x_1 = x_{\text{infl}} - \text{sign}(y_{\text{infl}})\frac{2}{3}\sqrt{D}$

repeat
  calculate polynomial value and derivatives, y, y', y''
  $\Delta x := \frac{yy'}{y'^2 - 0.5yy''}$
  $x_1 := x_1 - \Delta x$
until $|\Delta x| \le 10^{-6}|x_1|$
if $D > 0$ **then**
  $\text{Deflate}(\vec{a}; \vec{c})$
  $n := \text{SolveQuadratic}(c_0, c_1, 1; x_2, x_3)$
  if $n > 0$ **then** sort $\vec{x}$

  return $n + 1$
end
return 1

$\text{Deflate}(\vec{a}, x_1; \vec{b})$—an implementation of Eq. (24)—computes the coefficients $\vec{b}$ of the quadratic polynomial that results from dividing the cubic polynomial $\vec{a}$ by $(x - x_1)$.
$\text{SolveQuadratic}(a_0, a_1, a_2; x_1, x_2)$—an implementation of Eq. (25)—computes the roots of a normalized quadratic polynomial with the coefficients a_0, a_1, a_2, sorts them and returns the number of real roots.

Figure 3. Algorithm outlining the proposed iterative root finder for cubic polynomials.

The test runs were carried out on various PCs and servers having processors with x86-64 architecture, using a Linux operating system and the GNU C++ compiler g++-4.7 with the option "-Ofast". The CPU times reported in this work are for an Intel i5-2500K processor (3.3 GHz clock frequency). The results for other processor types (AMD Opteron 6218 (2.0 GHz), Intel Xeon E5-2650 (2.0 GHz)) were practically the same after compensating for their lower clock frequencies.

No runs were made for cases 1.4, 2.*x*, and 3.2, as these can be solved quicker analytically. These cases can be recognized by evaluating the value and the first derivative of the polynomial at the inflection point, $f(x_{\text{infl}})$ and $f'(x_{\text{infl}})$.

In order to demonstrate the effect of scaling, we created some "unscaled" coefficient sets by setting $A_i' = A_i\lambda^{3-i}$ with $\lambda = 10^3$ or 10^{-3}. This corresponds to a change of units for volumes from dm^3 to m^3 or to cm^3, respectively . The results are shown in Table 3.

From Table 3 it is evident that scaling can significantly speed up the computation if Cauchy bounds are used, whereas there is no perceivable improvement in the case of Laguerre−Nair−Samuelson bounds. If these bounds are used, it is better not to scale the polynomial in order to avoid the computational overhead of his operation.

Vieta initialization can be very fast, but also very slow, depending on the values of the polynomial coefficients. As it is possible to construct polynomials for which the Vieta initialization will led to divergent behavior, we have excluded it from further consideration.

Another interesting option is working with a shifted polynomial:

$$\Delta x^3 + b_1\Delta x + b_0 = 0 \tag{28}$$

with

$$\Delta x = x - x_{\text{infl}}$$

$$b_0 = a_0 + x_{\text{infl}}a_1 - 2x_{\text{infl}}{}^3$$

$$b_1 = x_{\text{infl}}a_2 + a_1$$

dx.doi.org/10.1021/ie4038664 | Ind. Eng. Chem. Res. 2014, 53, 2529−2536

As the shifted polynomial has no quadratic term anymore, the computation of function values and criteria is simplified. It turned out, however, that the CPU time savings could not compensate the computational overhead of the shift operation.

Table 4 contains the CPU times obtained for different iteration methods (scaling turned off, initialization with Laguerre−Nair−Samuelson bounds or inflection point) and, for comparison, for the analytical method (Cardano's formula). In the latter, the calculation of the roots was followed by one step of a Newton−Raphson iteration to reduce rounding errors, and by sorting. It turns out that the following are true:

(i) The analytical method is reasonably fast if there is only one real root, but rather slow otherwise.

(ii) Some of the oldest iteration methods—Newton−Raphson and Halley—perform best, with the latter being marginally better.

(iii) The speed of the regula falsi method depends very much on the polynomial coefficients. It can be extremely fast, but it is usually is slower than the other iterative methods.

This assessment of iterative methods is based on CPU times. Alternatively, one might prefer to look at the number of iteration steps required to obtain a given precision. For the test problems of this article, Newton's method needs two to three steps to achieve the desired precision if there are three real roots, and six otherwise. Higher-order methods (Halley, Olivera-Fuentes, modified Richmond) need two to three steps (three roots) or four steps (one root). Steffensen's methods is in between (three to four or five steps, respectively). The regula falsi method, having a convergence order of about 1.6, needs 4−10 steps (three roots) or 12−15 steps (one root). [If x_1, x_2, ..., x_∞ are successive approximations of an iteration method, the convergence order C of a convergent series is defined such that $\lim_{i\to\infty} |x_{i+1} - x_\infty|/|x_i - x_\infty|^C = \lambda$ is a positive constant.]

We conclude that—for the coefficient sets and test conditions used here—an algorithm consisting of

1. normalization
2. initialization with either Laguerre−Nair−Samuelson bounds or the inflection point
3. iterative search for the first root with Halley's method
4. deflation and solving the resulting quadratic equation analytically

(cf. algorithm in Figure 3) is usually best. [For convenience, a C/C++ subroutine is provided in the Supporting Information .] It is always significantly faster than the analytical method. If there are three real roots, which is typical for vapor pressure calculations, the new method is faster by a factor of 8−10.

■ ASSOCIATED CONTENT

Supporting Information

Cubic root finder using Halley's method: C/C++ program code. C/C++ code snippet: normalization and scaling. This material is available free of charge via the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: ulrich.deiters@uni-koeln.de.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

R.M.-S. gratefully acknowledges the Instituto Politécnico Nacional for providing financial support for this work.

■ SYMBOLS

Symbols Referring to Equations of State

a = attraction parameter of an equation of state
b = volume parameter of an equation of state
p = pressure
R = universal gas constant
T = temperature
T_c = critical temperature
V_m = molar volume
ρ = molar density, $\rho = 1/V_m$

Superscripts

g = gas, vapor state
l = liquid

Symbols Referring to Polynomials and Polynomial Root Finders

A_i = coefficient of the ith power term of a general third-order polynomial ($A_3 \neq 1$)
a_i = coefficient of a normalized third-order polynomial ($a_3 = 1$)
b_i = coefficient of a scaled or shifted third-order polynomial
c_i = coefficient of a second-order polynomial
C = convergence order
D = discriminant of a quadratic polynomial
r = radius of the root circle of a polynomial

Subscripts

infl = inflection point
lower = Laguerre−Nair−Samuelson bound
upper = Laguerre−Nair−Samuelson bound
NR = Newton−Raphson method
sc = scaled polynomial

■ REFERENCES

(1) Redlich, O.; Kwong, J. N. S. On the thermodynamics of solutions. V. An equation of state—fugacities of gaseous solutions. *Chem. Rev.* **1949**, *44*, 233.

(2) Peng, D. Y.; Robinson, D. B. A new two-constant equation of state. *Ind. Eng. Chem. Fundam.* **1976**, *15*, 59.

(3) Salim, P. H.; Trebble, M. A. A modified Trebble−Bishnoi equation of state: thermodynamic consistency revisited. *Fluid Phase Equilib.* **1991**, *65*, 59.

(4) Ahlers, J.; Gmehling, J. Development of an universal group contribution equation of state. I: Prediction of liquid densities for pure compounds with a volume translated Peng−Robinson equation of state. *Fluid Phase Equilib.* **2001**, *191*, 177.

(5) Schmid, B.; Gmehling, J. Revised parameters and typical results of the VTPR group contribution equation of state. *Fluid Phase Equilib.* **2006**, *317*, 110.

(6) Avaullee, L.; Trassy, L.; Neau, E.; Jaubert, J.-N. Thermodynamic modeling for petroleum fluids I. Equation of state and group contribution for the estimation of thermodynamic parameters of heavy hydrocarbons. *Fluid Phase Equilib.* **1997**, *139*, 155.

(7) Jaubert, J.-N.; Privat, R.; Mutelet, F. Predicting the phase equilibria of synthetic petroleum fluids with the PPR78 approach. *AIChE J.* **2010**, *56*, 3225.

(8) Qian, J.-W.; Privat, R.; Jaubert, J.-N. Predicting the phase equilibria, critical phenomena and mixing enthalpies of binary aqueous systems containing alkanes, cycloalkanes, aromatics, alkenes and gases (N_2, CO_2, H_2S, H_2) with the PPR78 equation of state. *Ind. Eng. Chem. Res.* **2013**, *52*, 16457.

(9) Quiñones-Cisneros, S. E.; Deiters, U. K. An efficient algorithm for the calculation of phase envelopes of fluid mixtures. *Fluid Phase Equilib.* **2012**, *329*, 22.
(10) Deiters, U. K. The calculation of densities from cubic equations of state. *AIChE J.* **2002**, *48*, 882.
(11) Olivera-Fuentes, C. The optimal solution of cubic equations of state. *Lat. Am. Appl. Res.* **1993**, *23*, 243.
(12) Deiters, U. K. The calculation of densities from cubic equations of state [Reply to a letter to the Editor by H. Salim]. *AIChE J.* **2005**, *51*, 3310.
(13) Mollerup, J. *Thermodynamic Properties from a Cubic Equation of State (SEP 8601)*; Technical report; Institute of Chemical Engineering, Danish Technical University: Lyngby, 1986.
(14) Edmister, W. C.; Lee, B. I. *Applied Hydrocarbon Thermodynamics*, 2nd ed.; Gulf Publishing Co.: Houston, TX, 1984; Vol. 1.
(15) Jordan-Engeln, G.; Reutter, F. *Numerische Mathematik für Ingenieure*; Bibliographisches Institut: Mannheim, Germany, 1985.
(16) Muller's method. http://en.wikipedia.org/wiki/Muller's_method.
(17) Durand–Kerner method. http://en.wikipedia.org/wiki/Durand-Kerner_method.
(18) Deiters, U. K.; Kraska, Th. *High-Pressure Fluid Phase Equilibria—Phenomenology and Computation*; Kiran, E., Ed.; Supercritical Fluid Science and Technology 2; Elsevier: Amsterdam, 2012.
(19) Deiters, U. K. ThermoC project home page: http://thermoc.uni-koeln.de/index.html.