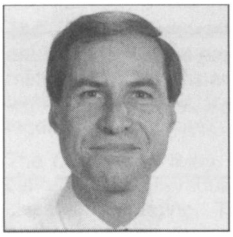
How (Not) to Solve Quadratic Equations

Yves Nievergelt



Yves Nievergelt (ynievergelt@ewu.edu) completed his diploma in mathematics from the École Polytechnique Fédérale de Lausanne in December of 1976, then earned a Ph.D. in several complex variables under the guidance of James R. King at the University of Washington in 1984. Since 1985, he has been teaching complex analysis and numerical analysis at Eastern Washington University, where he likes to emphasize the precise connections and mutual influences between real applications and abstract theories in mathematics.

Introduction

An 18th century method to solve quadratic equations proves more accurate than modern textbook methods, which can produce wrong magnitudes, wrong signs, and hence wrong conclusions. With its proven accuracy, this method reveals an error in the US Internal Revenue Service (IRS) documentation, and an application to chemistry shows that the accuracy of the data can be irrelevant.

MAA journals have already outlined the derivation of many methods [**2**], [**30**], machine arithmetic [**10**], [**27**], and their effect on rounding errors [**4**], [**11**]. Therefore, the present considerations restrict themselves to examples demonstrating how the situation can be worse than previously indicated in the literature.

To this end, each binary ("two-variable") arithmetic operation subject to rounding errors is denoted by a non-empty symbol, and the same symbol in a circle denotes the corresponding machine operation. For instance, $*$ denotes the mathematical multiplication, and $\circledast$ denotes the machine multiplication.

History

Two-millenia old methods to solve quadratic equations of the type

$$a * z^2 + b * z + c = 0 \tag{1}$$

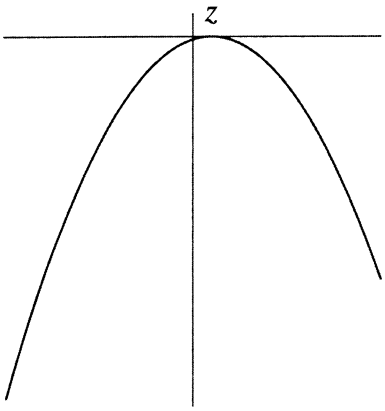
can be traced to purely algebraic and geometric problems in Babylonian and Greek texts, and to word problems with measurements in the Chinese "Nine Chapters on the Mathematical Art" [**37**]. Derived from completing the square [**37**, p. 52, 71], or from factoring given sums and products [**37**, p. 62–63], such methods are algebraically equivalent to the **usual quadratic formulae**

$$z_{\pm} = \frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}. \tag{2}$$

With only $2 * a$ in their denominator, formulae (2) simplify the division for computations by hand, especially for the applied or pedagogically contrived cases with a

Vol. 34, No. 2, March 2003

THE COLLEGE MATHEMATICS JOURNAL



$$-0.158114166017 * z^2 + 0.316227766017 * z - 0.1581136 = 0$$

METHOD	COMPUTED ROOT $\tilde{z}_1$	COMPUTED ROOT $\tilde{z}_2$
1	−0.0000017899	0.999998210097
2	−9.4347843986	0.999998210097
3	−9.4347843986	0.533139420472
4	−4.5000017899	1.00000000044
5	0.999996548529	0.999999963595

A Hard Equation

IN THIS ISSUE:

- You think you know how to solve quadratics?
- 2, 4, 8, 16: you think you know what comes next?
- Lissajous figures are Chebyshev polynomials
- Twelve minimum problems from one parabola

An Official Publication of the Mathematical Association of America

single-digit value for a, whereas square-roots could be read off tables. This advantage explains the lack of popularity of algebraically equivalent formulae derived by **Conte di Fagnano** in the 18th century [**6**, p. 415, eq. (6)]:

$$z_{\pm} = \frac{2 * c}{-b \mp \sqrt{b^2 - 4 * a * c}}. \tag{3}$$

Indeed, two centuries after Fagnano, the usual formulae (2) were still used on the DEUCE computer [**8**, p. 100]. However, the **modified Fagnano formulae**

$$z_1 = \frac{-2 * c}{b + \text{sign}(b)\sqrt{b^2 - 4 * a * c},} \tag{4}$$

$$z_2 = \frac{b + \text{sign}(b)\sqrt{b^2 - 4 * a * c}}{-2 * a,} \tag{5}$$

[where $\text{sign}(b) = 1$ if $b \geq 0$, and $\text{sign}(b) = -1$ otherwise] soon proved more accurate than (2) under the effect of computer rounding errors [**25**]. Similar to the trigonometric formulae for cubic equations [**7**, p. 35], there also exist trigonometric formulae for quadratic equations [**2**]: if $b^2 \geq 4 * a * c$, then

$$\alpha = \begin{cases} \arg\left(-b, 2\sqrt{|a * c|}\right) & \in \;]0, \pi[& \text{if } a * c < 0, \\ \arcsin\left(-2\sqrt{|a * c|}/b\right) & \in \; [-\pi/2, \pi/2] & \text{if } a * c > 0; \end{cases} \tag{6}$$

$$z_1 = \sqrt{|c/a|} \cdot \cot(\alpha/2), \tag{7}$$

$$z_2 = \sqrt{|c/a|} \cdot \tan(\alpha/2) \cdot \text{sign}(a * c), \tag{8}$$

whereas if $b^2 < 4 * a * c$, then

$$\alpha = \arccos\left[-b/\left(2 * \sqrt{|a * c|}\right)\right] \in]0, \pi[, \tag{9}$$

$$z_{\pm} = \sqrt{c/a} \cdot [\cos(\alpha) \pm i \sin(\alpha)]. \tag{10}$$

However, trigonometric functions can cause rounding errors larger than square roots can [**17**, p. 179], as corroborated by Table 7 in example 10, on page 101. Therefore, the state-of-the-art method will use formulae like (4)–(5).

The State of the Art

Algorithm 1 below relies on the **numerical analysts' quadratic formulae**:

$$z_1 = \frac{-c}{(b/2) + \text{sign}(b)\sqrt{(b/2)^2 - a * c},} \tag{11}$$

$$z_2 = \frac{(b/2) + \text{sign}(b)\sqrt{(b/2)^2 - a * c}}{-a.} \tag{12}$$

Formulae (11)–(12) involve only one division ($b/2$) whereas formulae (4)–(5) involve two multiplications ($2 * a$ and $2 * c$), but they yield the same results in binary arithmetic, where multiplications and divisions by 2 are exact.

Algorithm 1. For all non-zero reals a, b, c, compute z_1 and z_2 as follows.

$$h = b/2, \tag{13}$$

$$d^2 = h^2 - (a * c). \tag{14}$$

Thereafter, if $d^2 \geq 0$ (and a, h, c are all non-zero real numbers), then compute

$$d = \sqrt{d^2}, \tag{15}$$

$$t = -\left[h + \operatorname{sign}(h)d\right], \tag{16}$$

$$z_1 = c/t, \tag{17}$$

$$z_2 = t/a. \tag{18}$$

Otherwise, if $d^2 < 0$ (and a, h, c are all non-zero real numbers), then compute

$$x = -h/a, \tag{19}$$

$$y = \sqrt{-d^2}/a, \tag{20}$$

$$z_1 = x + iy, \tag{21}$$

$$z_2 = x - iy. \tag{22}$$

Special cases handle the situations where $a = 0$ or $b = 0$ or $c = 0$. ■

As a consequence, z_1 is the solution with the smaller magnitude, because

$$|z_1| = \left|\frac{|h| - |d|}{a}\right| \leq \left|\frac{|h| + |d|}{a}\right| = |z_2|. \tag{23}$$

However, example 2 will show that rounding errors can affect the computed solutions $\tilde{z}_1$ and $\tilde{z}_2$ so much as to reverse their order, so that $|\tilde{z}_2| < |\tilde{z}_1|$ while $|z_1| < |z_2|$. About such errors, David Goldberg proves the following result:

> *If $b^2 \approx 4ac$, rounding error can contaminate up to half the digits in the roots computed with the quadratic formula* $[-b \pm \sqrt{b^2 - 4ac}]/2a$.—D. Goldberg [**13**, p. 41].

Yet new theory shows that computations carrying m digits might yield only $k \geq (m/2) - 1$ accurate digits, fewer than Goldberg's $m/2$ [**26**]. Similarly, William M. Kahan states the following result for the special case with a coefficient $b = -2 * B$ that is an even integer, and with computations carrying $m = 10$ decimal digits:

> Consequently, the computed roots match the given quadratic's roots to at least five significant digits. More generally, if the roots x and y agree to n significant digits for some positive $n \leq 5$, then they are correct to at least $10 - n$ significant digits unless overflow or underflow occurs.—W. M. Kahan [**17**, p. 205].

Still, new theory also shows that computations carrying m digits might yield only $k \geq m - (n + 2)$ accurate digits, fewer than Kahan's $m - n$ [**26**].

Example 2. For the quadratic equation

$$-0.158114166017 * z^2 + 0.316\,227\,766\,017 * z - 0.158\,113\,600\,000 = 0, \tag{24}$$

$a + b + c = 0$ whence $z_2 = 1$ is a solution, and $z_1 = c/a = 0.999\,996\,420\,201\ldots$ is the other solution. Computations carrying $m = 12$ decimal digits produce the following results. (The "tilde" $\tilde{}$ denotes computed quantities as opposed to mathematical quantities, where underlined digits are incorrect.)

$$\tilde{h} = b \oslash 2 = 0.158\,113\,883\,008, \tag{25}$$

$$\widetilde{\tilde{h}^2} = \tilde{h} \circledast \tilde{h} = 0.024999\,999\,999\,9, \tag{26}$$

$$a \circledast c = 0.024999\,999\,999\,9, \tag{27}$$

$$\tilde{d}^2 = (\tilde{h} \circledast \tilde{h}) \ominus (a \circledast c) = 0, \tag{28}$$

$$\tilde{t} = -(\tilde{h} \oplus \tilde{d}) = -0.158\,113\,883\,008, \tag{29}$$

$$\tilde{z}_1 = c \oslash \tilde{t} = 0.999\,99\underline{8\,210\,100}, \tag{30}$$

$$\tilde{z}_2 = \tilde{t} \oslash a = 0.999\,99\underline{8\,210\,097}. \tag{31}$$

From $\tilde{d} = 0 < d$ followed $|\tilde{t}| < |t|$, whence $\tilde{z}_2 = \tilde{t} \oslash a < z_2$ and $\tilde{z}_1 = c \oslash \tilde{t} > z_1$ with errors so large that $\tilde{z}_1 > \tilde{z}_2$ whereas $z_1 < z_2$. Moreover, the computed solutions $\tilde{z}_1$ and $\tilde{z}_2$ differ from the real solutions z_1 and z_2 by more than one unit in their 6th significant digit. Thus they are accurate to only $k = 5$ significant digits—as predicted by the theory that $k = 5 = (12/2) - 1 = (m/2) - 1$ in the worst case [**26**]—but one digit worse than in Goldberg's theorem. Furthermore, the real solutions z_1 and z_2 differ from each other by about one half of one unit in their sixth significant digit, so that they agree with each other to $n = 5$ digits—also corroborating the theory that $k = 5 = 12 - (5 + 2) = m - (n + 2)$ in the worst case [**26**]—but two digits worse than in Kahan's result with even coefficients. ■

Optional Extended Precision

Facilities for extended precision accumulations of dot-products have long been available on a variety of machines, from ACE in the 1950s [**38**, p. 27] to Hewlett-Packard's HP-15C, HP-28 and HP 48 series calculators in the 1980s [**17**, p. 208], [**18**, p. 14-17]. As James Hardy Wilkinson pointed out, the use of such facilities "may be of vital importance to the accuracy of the final answers" [**38**, p. 32].

Example 3. For the quadratic equation

$$-0.3\,124\,999\,999\,99 * z^2 + 0.707\,106\,781\,186 * z - 0.4 = 0, \tag{32}$$

integer calculations with *Mathematica* [**41**] produce a positive discriminant,

$$d^2 = b^2 - 4 * a * c = 825\,683\,566\,596 * 10^{-24} > 0, \tag{33}$$

whence $d = 9.086713193\ldots * 10^{-7}$ and equation (32) has two real solutions:

$$z_1 = 1.131\,369\,396\,027\ \ldots \tag{34}$$

$$z_2 = 1.131\,372\,303\,775\ \ldots \tag{35}$$

Yet computations with twelve decimal digits yield different results:

$$b \circledast b = 0.707\,106\,781\,1 \circledast 0.707\,106\,781\,1 = 0.499\,999\,999\,999; \tag{36}$$

$$a \circledast c = 0.3\,124\,999\,999 \circledast 0.4 = 0.125\,000\,000\,000; \tag{37}$$

$$4 \circledast (a \circledast c) = 4 \circledast 0.125\,000\,000\,0 = 0.500\,000\,000\,000; \tag{38}$$

$$\tilde{d}^2 = (b \circledast b) \ominus [4 \circledast (a \circledast c)] \tag{39}$$

$$= -0.000\,000\,000\,001 < 0. \tag{40}$$

The inequality $\tilde{d}^2 < 0$ would indicate that equation (32) has two complex solutions, which—with real arithmetic—forces the use of formulae (2):

$$\tilde{z}_{\pm} = 1.131\,37\underline{0\,849\,9} \pm i * \underline{1.600\,000\,000\,01 * 10^{-6}}. \tag{41}$$

The computed solutions $\tilde{z}_+$ and $\tilde{z}_-$ agree with the solutions z_1 and z_2 to only five significant digits. In contrast, one extended-precision dot-product restores all but the last significant digit. For instance, the HP-28 & 48 series compute dot-products internally with 15 digits [**18**, pp. 14–17]. To take advantage of it, use (4)–(5) and re-write the discriminant as a dot-product:

$$b^2 - 4ac = (b, -a, -a, -a, -a) \cdot (b, c, c, c, c). \tag{42}$$

Using the dot-product command (`A B DOT`) produces the result

$$\mathtt{A} \cdot \mathtt{B} = 8.25 * 10^{-13}, \tag{43}$$

which is correct to two digits. Hence the computations resume with ordinary calculator operations, with `float` ($\sqrt{\ }$) denoting the computed value of $\sqrt{\ }$:

$$\tilde{d} = \mathtt{float}\left(\sqrt{8.25 * 10^{-13}}\right) = 9.082\,951\,062\,29 \cdot 10^{-7}, \tag{44}$$

$$\tilde{t} = -\left[b \oplus \operatorname{sign}(b) d\right] = -0.707\,107\,689\,481, \tag{45}$$

$$\tilde{z}_1 = t \oslash (2 \circledast a) = \frac{0.707\,107\,689\,481}{2 * 0.3\,124\,999\,999} = 1.131\,369\,396\,\underline{63}; \tag{46}$$

$$\tilde{z}_2 = (2 \circledast c) \oslash t = \frac{2 * 0.4}{0.707\,107\,689\,481} = 1.131\,372\,303\,\underline{17}, \tag{47}$$

which are now correct to ten significant digits. Table 1 corroborates the theory: the last two rows show that the two real solutions agree with each other to $n = 6$ significant digits, and the first two rows show that computations carrying $m = 12$ digits produce only to $k = 5$ correct significant digits, confirming that $(m/2) - 1 = k = m - (n + 2)$ in the worst case [**26**]. Again, (11)–(12) yield a greater accuracy than (4)–(5). The greater accuracy with equivalent integer coefficients is consistent with initial rounding errors in the conversion from decimal data to their internal binary representations. ■

Comparisons of Algorithms

Some texts recommend computing first z_2, and then $z_1 = c/(a * z_2)$; see [**15**, p. 43], [**16**, p. 13], [**19**, p. 11], [**21**, p. 24], [**32**, p. 79], [**33**, p. 248], [**34**, p. 21]. However,

Table 1. Ordinary (2), (11)–(12), and extended-precision operations (`DOT`, `Solve poly`, `roots`, `Root`).

<table>
<tr><td colspan="4">$-0.3\,124\,999\,999\,99 * z^2 + 0.707\,106\,781\,186 * z - 0.4 = 0$</td></tr>
<tr><td>SYSTEM[a]</td><td>FORMULAE</td><td>$\tilde{z}_1$</td><td>$\tilde{z}_2$</td></tr>
<tr><td>HP 48GX</td><td>(2)</td><td>$1.1313\underline{708499}$
$+i * \underline{1.600\,000\,000\,01 \cdot 10^{-6}}$</td><td>$1.1313\underline{708499}$
$-i * \underline{1.600\,000\,000\,01 \cdot 10^{-6}}$</td></tr>
<tr><td>HP 48GX</td><td>(11)–(12)</td><td>$1.1313\underline{708499}$</td><td>$1.13137\underline{08499}$</td></tr>
<tr><td>HP 48GX, DOT</td><td>(4)–(5)</td><td>$1.131369396\underline{63}$</td><td>$1.131372303\underline{17}$</td></tr>
<tr><td>HP 48GX</td><td>Solve poly</td><td>$1.131369396\underline{63}$</td><td>$1.131372303\underline{17}$</td></tr>
<tr><td>MATLAB</td><td>(11)–(12)</td><td>$1.1313693960\underline{8541}$</td><td>$1.1313723037\underline{1703}$</td></tr>
<tr><td>MATLAB</td><td>roots</td><td>$1.131369396\underline{14133}$</td><td>$1.131372303\underline{66111}$</td></tr>
<tr><td><i>Mathematica</i></td><td>(11)–(12)</td><td>$1.131369396\underline{085406}$</td><td>$1.131372303\underline{717035}$</td></tr>
<tr><td><i>Mathematica</i></td><td>Root</td><td>$1.13136939596\underline{6732}$</td><td>$1.131372303\underline{835709}$</td></tr>
<tr><td colspan="4">$-312\,499\,999\,999 * z^2 + 707\,106\,781\,186 * z - 400\,000\,000\,000 = 0$</td></tr>
<tr><td>MATLAB</td><td>(11)–(12)</td><td>$1.1313693960\underline{0607}$</td><td>$1.1313723037\underline{9637}$</td></tr>
<tr><td>MATLAB</td><td>roots</td><td>$1.131369396\underline{14133}$</td><td>$1.131372303\underline{66111}$</td></tr>
<tr><td><i>Mathematica</i></td><td>(11)–(12)</td><td>1.1313693960271 . . .</td><td>1.131372303775 . . .</td></tr>
<tr><td><i>Mathematica</i></td><td>Root</td><td>1.1313693960271 . . .</td><td>1.131372303775 . . .</td></tr>
</table>

[a] *Mathematica 3.0* & `MATLAB 5.2` computations with a Power Macintosh G3.

because the computation of $z_2 = t/a$ already requires the computation of t, the computation $z_1 = c/(a * z_2)$ mars the result with rounding errors from two arithmetic operations more than does $z_1 = c/t$, which is also the computation recommended in [**3**, p. 12], [**14**, p. 17], [**16**, p. 51], [**20**, p. 17].

Other texts recommend performing the transformation $w = 1/z$, then computing $\tilde{w}_1$ or $\tilde{w}_2$, depending on whether the available methods favor the smaller [**22**, p. 9] or the larger [**32**, p. 79] solution, and then computing the inverse $\tilde{z}_k = 1 \oslash \tilde{w}_k$ [**1,** p. 185]. Such inversions cause the same and more errors, because $w_2 = t/c$ and $z_1 = 1/(w_2)$, and likewise $w_1 = t/a$ and $z_2 = 1/(w_1)$.

Example 4. For the quadratic equation

$$-67 * z^2 + 134 * z - 65 = 0, \tag{48}$$

Table 2 confirms that carrying $m = 4$, or $m = 10$, or $m = 12$, digits, algorithm 1 is more accurate than the computations $\tilde{z}_1 = c \oslash (a \circledast \tilde{z}_2)$ and $\tilde{z}_1 = 1 \oslash \tilde{w}_2$. ■

Some software packages compute polynomial roots by computing the eigenvalues of a companion matrix, for instance, with J. F. G. Francis's QR method [**34**], [**40**], or its QZ variant, as with MATLAB's `roots` [**23**], but not so accurately:

> However, MATLAB and LAPACK could be more accurate! Both packages compute the Schur form of a 2 × 2 matrix using an algorithm that is more unstable for the smaller eigenvalue than is necessary. We propose that such high-quality packages should compute the eigenvalues of a general 2 × 2 matrix by solving the quadratic equation as accurately as possible, given the rounded values of the trace and the determinant. If we have a 2 × 2 companion matrix, then there will be no roundoff error in the trace and the determinant.—A. Edelman and H. Murakami [**9**, p. 769].

Table 2. Inaccuracies from $c \oslash (a \circledast \tilde{z}_2)$ and $1 \oslash \tilde{w}_2$.

$-67 * z^2 + 134 * z - 65 = 0$				
METHOD	ITEM	4 digits	HP-15C	HP 48GX
Algorithm 1	$\tilde{z}_2$	1.173	1.172 773 685	1.172 773 685 12
Algorithm 1	$\tilde{z}_1$	0.827 2	0.827 226 314 9	0.827 226 314 884
$c \oslash (a \circledast \tilde{z}_2)$	$\tilde{z}_1$	$0.827\,\underline{1}$	0.827 226 314 9	$0.827\,226\,314\,88\underline{2}$
$-67.00 + 134.0 * w - 65.00 * w^2 = 0$				
Algorithm 1	$\tilde{w}_2$	1.209	1.208 859 029	1.20885902927
Algorithm 1	$\tilde{w}_1$	0.852 6	0.852 679 432 3	0.852 679 432 265
$1 \oslash \tilde{w}_1$	$\tilde{z}_2$	1.173	1.172 773 685	1.172 773 685 12
$1 \oslash \tilde{w}_2$	$\tilde{z}_1$	$0.827\underline{1}$	$0.827\,226\,31\underline{5\,1}$	$0.827\,226\,314\,88\underline{6}$
Mathematica `Root`	$\tilde{z}_2$	1.17277368511627202199337030017843065353880		
Mathematica `Root`	$\tilde{z}_1$	0.82722631488372797800662969982156934646119		

However, companion matrices apply only to monic polynomials [**9**], [**35**]. In general, the division by the leading coefficient introduces rounding errors in the trace and determinant of the companion matrix before the computations start:

$$\begin{vmatrix} 0 & -c/a \\ 1 & -b/a \end{vmatrix} \neq \begin{vmatrix} 0 & -c \oslash a \\ 1 & -b \oslash a \end{vmatrix}. \tag{49}$$

Example 5. For the same equation (24) as in example 2, Table 3 shows that algorithm 1 is more accurate than the QZ method in all cases. ■

Table 3. Inaccuracies from the QZ method.

$-0.158\,114\,166\,017 * z^2 + 0.316\,227\,766\,017 * z - 0.158\,113\,600\,000 = 0$			
SYSTEM[a]	METHOD	$\tilde{z}_1$	$\tilde{z}_2$
HP 48GX[b]	(4)–(5), (42)	$0.99999642\underline{125}$	$0.99999999\underline{895}$
HP 48GX	`Solve poly`	$0.99999642\underline{1529}$	$0.99999999\underline{8672}$
`MATLAB`	Algorithm 1	$0.9999964202\underline{1877}$	$0.9999999999\underline{8181}$
`MATLAB`	`roots` (QZ)	$0.9999964202\underline{6261}$	$0.9999999999\underline{3797}$
Mathematica	Algorithm 1	$0.99999642021\underline{877}$	$0.9999999999\underline{8181}$
Mathematica	`Root`	$0.9999964202\underline{8509}$	$0.9999999999\underline{1549}$
$-158\,114\,166\,017 * z^2 + 316\,227\,766\,017 * z - 158\,113\,600\,000 = 0$			
`MATLAB`	Algorithm 1	$0.9999964201\underline{9367}$	$1.0000000000\underline{0691}$
`MATLAB`	`roots` (QZ)	$0.9999964202\underline{3159}$	$0.9999999999\underline{6899}$
Mathematica	Algorithm 1	0.99999642020058	1.00000000000000
Mathematica	`Root`	0.99999642020058	1.00000000000000
Solutions	Algebra	0.99999642020057874 . . . 1	

[a] *Mathematica 3.0* & `MATLAB 5.2` computations with a Power Macintosh G3.
[b] With extended-precision dot products.

Newton's method is also recommended to improve the accuracy of computed solutions of equations [**16**, p. 95]. Nevertheless, Wilkinson had already identified that a difficulty with iterative refinements lies in the evaluation of functions:

The ‘difficulty’ with the polynomial [. . .] is that of evaluating the *explicit* polynomial accurately.—J. H. Wilkinson [**39**, p. 14].

Example 6. Continuing examples 2 and 5, this example shows how Newton’s method can destroy all of the existing accuracy, including the magnitude and the sign, of the initial computed solutions. To this end, define

$$\begin{aligned} Q(z) = {} & -0.158\,114\,166\,017 * z^2 \\ & + 0.316\,227\,766\,017 * z - 0.158\,113\,600\,000, \end{aligned} \tag{50}$$

and let $\tilde{Q}(z)$ and $\widetilde{Q'}(z)$ denote the computed values of $Q(z)$ and $Q'(z)$. A recommended evaluation procedures utilizes Horner’s algorithm [**16**, p. 95]:

$$\tilde{Q}(z) = \{[(a \circledast z) \oplus b] \circledast z\} \oplus c, \tag{51}$$

$$\widetilde{Q'}(z) = [2 \circledast (a \circledast z)] \oplus b, \tag{52}$$

which also lends itself to extended-precision dot-products:

$$\tilde{Q}_3(\tilde{z}_{1,0}) = \left\{\left[(\tilde{z}_{1,0}, 1) \cdot (a, b)\right], c\right\} \cdot (\tilde{z}_{1,0}, 1), \tag{53}$$

$$\widetilde{Q'_3}(\tilde{z}_{1,0}) = (\tilde{z}_{1,0}, \tilde{z}_{1,0}, 1) \cdot (a, a, b). \tag{54}$$

Another recommended evaluation procedure uses any available facilities to solve linear systems with an extended precision [**19**, p. 108], [**24**, p. 32]. Specifically, $\tilde{Q}(z)$ and $\tilde{Q}'(z)$ are components of the solutions of the following linear systems:

$$\begin{pmatrix} 1 & 0 & 0 \\ -z & 1 & 0 \\ 0 & -z & 1 \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \tag{55}$$

$$r_3 = \tilde{Q}(z); \tag{56}$$

$$\begin{pmatrix} 1 & 0 \\ -z & 1 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 2a \\ b \end{pmatrix}, \tag{57}$$

$$s_2 = \widetilde{Q'}(z). \tag{58}$$

For instance, the HP-15C, HP-28 & 48 series calculators solve linear systems with an extended precision. Alternatively, `MATLAB`’s `polyval` and the HP 48G series’s `PEVAL` evaluate polynomials in extended precision. Table 4 shows that $\tilde{Q}(z)$ and $\widetilde{Q'}(z)$ can still be so inaccurate that the first iteration $\tilde{z}_{k,1}$ of Newton’s method can destroy all accurate digits and sign in the initial solution $\tilde{z}_{k,0}$. ■

In examples 6, Newton’s method fails because the magnitudes of the rounding errors overwhelm the small magnitudes of $\tilde{Q}(z)$ and $\widetilde{Q'}(z)$. Therefore, some texts recommend methods of higher order [**1**, p. 186], [**22**, p. 11]. Yet approximations of higher orders to a quadratic polynomial coincide with the same quadratic polynomial: computing the roots of the approximating functions amounts to computing the roots of the same quadratic polynomial. Other texts recommend centering such methods at the ex-

Table 4. Worse errors after one step of Newton's method.

$-0.158\,114\,166\,017 * z^2 + 0.316\,227\,766\,017 * z - 0.158\,113\,600\,000 = 0$			
METHOD[a]	ITEM	FOR z_1	FOR z_2
Ordinary-precision calculator arithmetic			
Algorithm 1	$\tilde{z}_{k,0}$	$0.99999\underline{8210100}$	$0.99999\underline{8210097}$
Horner (51)	$\underset{\sim}{Q}(\tilde{z}_{k,0})$	0.000000000001	0.000000000000
Horner (52)	$\widetilde{Q}'(\tilde{z}_{k,0})$	0.000000000001	0.000000000001
$\div$	$\frac{\tilde{Q}(\tilde{z}_{k,0})}{\widetilde{Q}'(\tilde{z}_{k,0})}$	1.000000000000	0.000000000000
$-$	$\tilde{z}_{k,1}$	$\underline{-0.0000017899}$	$0.99999\underline{8210097}$
Extended-precision dot-product and ordinary calculator arithmetic			
Algorithm 1	$\tilde{z}_{k,0}$	$0.99999\underline{8210100}$	$0.99999\underline{8210097}$
DOT (53)	$\underset{\sim}{Q}(\tilde{z}_{k,0})$	$9.6 \cdot 10^{-13}$	0.000000000000
DOT (54)	$\widetilde{Q}'(\tilde{z}_{k,0})$	$9.2 \cdot 10^{-14}$	$1.042 \cdot 10^{-12}$
$\div$	$\frac{\tilde{Q}(\tilde{z}_{k,0})}{\widetilde{Q}'(\tilde{z}_{k,0})}$	10.4347826087	0.000000000000
$-$	$\tilde{z}_{k,1}$	$\underline{-9.4347843986}$	$0.99999\underline{8210097}$
Extended-precision polynomial evaluation with linear systems			
Algorithm 1	$\tilde{z}_{k,0}$	$0.99999\underline{8210100}$	$0.99999\underline{8210097}$
System (56)	$\underset{\sim}{Q}(\tilde{z}_{k,0})$	$9.6 \cdot 10^{-13}$	$4.86 \cdot 10^{-13}$
System (58)	$\widetilde{Q}'(\tilde{z}_{k,0})$	$9.2 \cdot 10^{-14}$	$1.041 \cdot 10^{-12}$
$\div$	$\frac{\tilde{Q}(\tilde{z}_{k,0})}{\widetilde{Q}'(\tilde{z}_{k,0})}$	10.4347826087	$0.466\,858\,789\,625$
$-$	$\tilde{z}_{k,1}$	$\underline{-9.4347843986}$	$0.\underline{533\,139\,420\,472}$
Extended-precision polynomial evaluation with "P EVAL" routine			
Algorithm 1	$\tilde{z}_{k,0}$	$0.99999\underline{8210100}$	$0.99999\underline{8210097}$
PEVAL	$\underset{\sim}{Q}(\tilde{z}_{k,0})$	$5.06 \cdot 10^{-13}$	$1 \cdot 10^{-15}$
PEVAL	$\widetilde{Q}'(\tilde{z}_{k,0})$	$9.2 \cdot 10^{-14}$	$-5.65597048 \cdot 10^{-7}$
$\div$	$\frac{\tilde{Q}(\tilde{z}_{k,0})}{\widetilde{Q}'(\tilde{z}_{k,0})}$	5.5	$-1.76804317409 \cdot 10^{-9}$
$-$	$\tilde{z}_{k,1}$	$\underline{-4.5000017899}$	$1.000000000\underline{44}$
Extended-precision initial values & polynomial evaluation by linear systems			
(4)–(5), (42)	$\tilde{z}_{k,0}$	$0.99999642\underline{125}$	$0.99999999\underline{895}$
System (56)	$\underset{\sim}{Q}(\tilde{z}_{k,0})$	$-7.2 \cdot 10^{-14}$	$-2. \cdot 10^{-14}$
System (58)	$\widetilde{Q}'(\tilde{z}_{k,0})$	$5.65685144 \cdot 10^{-7}$	$-5.6568496 \cdot 10^{-7}$
$\div$	$\frac{\tilde{Q}(\tilde{z}_{k,0})}{\widetilde{Q}'(\tilde{z}_{k,0})}$	$-1.27279283827 \cdot 10^{-7}$	$3.53553681187 \cdot 10^{-8}$
$-$	$\tilde{z}_{k,1}$	$0.999996\underline{548529}$	$0.9999999\underline{63595}$

[a] All computations with a Hewlett-Packard HP 48GX.

tremum between the two solutions, whence "Newton's method will work quite well" [**1**, p. 186], perhaps.

Example 7. Continuing examples 2, 5, 6, computations carrying 12 digits indicate that the extremum occurs at $\tilde{z}_{\max} = -b \oslash (2 \circledast a) = 0.999\,808\,510\,113$, where

$\tilde{Q}(\tilde{z}_{\max}) = -0.000\,029\,994\,202 < 0$, which would imply that the osculating quadratic polynomial—which coincides with the proposed polynomial—has no real solution. Moreover, this extremum is a maximum because $a < 0$, and on either side of each root the polynomial also takes on negative values. Consequently, such alternative iterative refinements as bisection or *regula falsi* [**34**, p. 307–308] can also fail to detect where the polynomial changes sign. ■

Applications

Some entire fields of applications involve only coefficients that satisfy certain relations, such as real coefficients with $a * c < 0$, and in such cases the theory shows that a calculator carrying m decimal digits with algorithm 1 produces computed solutions accurate to at least $k \geq m - 2$ digits [**26**].

Example 8. The Federal Reserve Bank of New York [**36**, p. 18] shows by an example how to compute the internal rate of return r of a US Treasury bill maturing after $D = 363$ days, in a calendar year with $Y = 365$ days, with a face-value $F = \$100$, and a purchase price $P = \$87.825$:

$$\begin{aligned} b = D/Y \quad &= \quad 363/365 &&= \ 0.994\,520\,547\,945\ldots, \\ a = (b/2) - (1/4) &= [363/(2 * 365)] - 0.25 &&= \ 0.247\,260\,273\,973\ldots, \\ c = (P - F)/P \quad &= (87.825 - 100)/87.825 &&= -0.138\,627\,953\,316\ldots. \end{aligned}$$

The internal rate of return r is then *defined* as the non-negative solution of

$$0.247\,260\,273\,973 * z^2 + 0.994\,520\,547\,945 * z - 0.138\,627\,953\,316 = 0. \quad (59)$$

After a multiplication by $4 * P * Y$ to get integer coefficients, the internal rate of return r is also the non-negative solution of

$$31704825 * z^2 + 127521900 * z - 17775500 = 0. \quad (60)$$

Table 5 corroborates such an accuracy with a ten-digit calculator, which produces $k = 9 > 8 = 10 - 2 = m - 2$ correct digits, and a twelve-digit calculator, which produces $k = 12 > 10 = 12 - 2 = m - 2$ correct digits. An eight-digit calculator without exponents TI-1706 II (the official calculator of the Casualty Actuarial Society and the Society of Actuaries) either truncates the coefficients of equation (59) to seven significant digits, keeping one digit for the leading zero, or requires a division of the integer coefficients of equation (60) by their greatest common divisor and then another scaling to fit the coefficients in the display. In either case, the calculator produces results accurate to at least $k = 6 = 8 - 2 = m - 2$ digits, as shown in Table 5. ■

In a different application, the US corporate tax code calls for the smaller non-negative solution of a quadratic equation [**28**], [**29**] whose coefficients satisfy the inequalities $0 \leq a * c \leq h^2$, which correspond to the same worst cases as in examples 2, 5, 6. Under the stronger inequalities $0 \leq a * c \leq h^2/2$, the theory also predicts that a calculator carrying m decimal digits with algorithm 1 produces computed solutions accurate to at least $k \geq m - 2$ digits [**26**].

Example 9. The US Internal Revenue Service (IRS) documentation [**29**, §1.861-12T, p. 203] shows an example of a US corporation X with total assets A and total

Table 5. Numerical computations of the internal rate of return $r = z_1$.

<table>
<tr><td colspan="4">$0.247\,260\,273\,973 * z^2 + 0.994\,520\,547\,945 * z - 0.138\,627\,953\,316 = 0$</td></tr>
<tr><td>SYSTEM[a]</td><td>FORMULAE</td><td>$\tilde{z}_1$</td><td>$\tilde{z}_2$</td></tr>
<tr><td>HP-15C</td><td>Algorithm 1</td><td>$0.134869362\underline{3}$</td><td>$-4.15703002\underline{5}$</td></tr>
<tr><td>HP 48GX</td><td>Algorithm 1</td><td>0.134869362221</td><td>−4.15703002704</td></tr>
<tr><td>HP 48GX</td><td>Solve poly</td><td>0.134869362221</td><td>−4.15703002703</td></tr>
<tr><td>MATLAB</td><td>Algorithm 1</td><td>0.13486936222116</td><td>−4.15703002704111</td></tr>
<tr><td>MATLAB</td><td>roots (QZ)</td><td>0.13486936222116</td><td>−4.15703002704111</td></tr>
<tr><td>*Mathematica*</td><td>Algorithm 1</td><td>0.1348693622211607</td><td>−4.157030027041104</td></tr>
<tr><td>*Mathematica*</td><td>Root</td><td>0.1348693622211607</td><td>−4.157030027041105</td></tr>
<tr><td colspan="4">$1.268193 * z^2 + 5.100876 * z - 0.711020 = 0$</td></tr>
<tr><td>HP-15C</td><td>Algorithm 1</td><td>0.1348693622</td><td>−4.157030027</td></tr>
<tr><td>TI-1706 II</td><td>Algorithm 1</td><td>$0.134869\underline{3}$</td><td>$-4.157029\underline{9}$</td></tr>
<tr><td colspan="4">$0.2472602 * z^2 + 0.9945205 * z - 0.1386279 = 0$</td></tr>
<tr><td>TI-1706 II</td><td>Algorithm 1</td><td>$0.134869\underline{3}$</td><td>$-4.157030\underline{1}$</td></tr>
</table>

[a] *Mathematica 3.0* & MATLAB 5.2 computations with a Power Macintosh G3.

debts D, which also controls a foreign subsidiary X' with total assets A' and total debts D'. If the allowed percentage p (80% after 1989) of the debt-to-asset ratio D/A of the US corporation exceeds that of the foreign subsidiary D'/A', then the IRS code defines the "excess related person indebtedness" [**28**] as the smaller non-negative solution Z of the equation

$$\frac{D-Z}{A-Z} \cdot p = \frac{D'+Z}{A'}, \tag{61}$$

$$Z^2 + [d - (A + pA')] * Z + (pDA' - AD') = 0. \tag{62}$$

Thus $a = 1$, and the condition that $pD/A > D'/A'$ is equivalent to $c > 0$, so that $0 < a * c$. If both debt-to-asset ratios remain sufficiently small, then also $a * c \leq h^2$. The IRS code gives a numerical example with $A = 2 \cdot 10^6$, $D = 10^6$, $A' = 5 \cdot 10^5$, $D' = 10^5$ and $p = 0.8$, which produces the coefficients $a = 1, b = D' - (A + pA') = -2\,300\,000$, $c = pDA' - AD' = 2 * 10^{11}$, and hence gives

$$z^2 - 2\,300\,000 * z + 2 * 10^{11} = 0. \tag{63}$$

Thus the inequality $a * c = 2 * 10^{11} < 6.6125 * 10^{11} = h^2/2$ holds. The IRS example gives the value 90 519 for the smaller solution z_1, and $2.21 * 10^{11}$ (sic) for z_2. An eight-digit calculator without exponents (TI-1706 II) requires a scaling of the integer coefficients of (63) to fit the coefficients in the display and prevent overflow and underflow. The calculator produces results accurate to all displayed digits, as do other systems, as shown in Table 6. Because the theory guarantees $k \geq m - 2$ correct digits [**26**], $\tilde{z}_2 = 2.209\ldots * 10^6$: the IRS solution $\tilde{z}_2 = 2.21 * 10^{11}$ must contain a typographical error in the exponent. ■

Yet other fields of applications involve only coefficients that satisfy other relations, such as real coefficients with $a * c < 0$ and $|a * c|$ "much smaller than" h^2, abbrevi-

Table 6. Numerical computations of the excess related person indebtedness.

$z^2 - 2\,300\,000 * z + 2 * 10^{11} = 0$			
SYSTEM[a]	FORMULAE	$\tilde{z}_1$	$\tilde{z}_2$
HP-15C	Algorithm 1	90518.99498	2209481.005
HP 48GX	Algorithm 1	90518.9949792	2209481.00502
HP 48GX	`Solve poly`	90518.9949791	2209481.00502
`MATLAB`	Algorithm 1	90518.99497914547	2209481.005020855
`MATLAB`	`roots` (QZ)	90518.99497915	2209481.00502085
Mathematica	Algorithm 1	90518.994979145...	2209481.005020854...
Mathematica	`Root`	90518.994979145...	2209481.005020854...
$0.000001 * z^2 - 2.3 * z + 200000 = 0$			
TI-1706 II	Algorithm 1	90518.995	2209481.

[a] *Mathematica 3.0* & `MATLAB 5.2` computations with a Power Macintosh G3.

ated by $|a * c| \ll h^2$, and in such cases the theory shows that algorithm 1 produces computed solutions accurate to the penultimate digit [**26**], so that $k \geq m - 1$, but other algorithms do not fare as well.

Example 10. Hydrofluoric acid HF partly dissociates in water into a hydrogen ion H^+ and a fluor ion F^-. From an initial concentration C, measured in moles [M] of HF per liter [ℓ] of water, only a fraction zC dissociates, with $0 \leq z \leq 1$. The fraction zC that dissociates depends on the ionization constant K and the initial concentration C through the quadratic equation [**12**, p. 186, eq. (73)]:

$$C * z^2 + K * z - K = 0. \tag{64}$$

At 25°C, $K = 6.76 * 10^{-4}$ [**31**, p. 351, table 23-1] (the number of digits listed here is not an indication of accuracy [**31**, p. 350]). With a concentration of $C = 10^{-15}[\mathrm{M}/\ell]$, the dissociated fraction z is the non-negative solution z of

$$10^{-15} * z^2 + (6.76 * 10^{-4}) * z - (6.76 * 10^{-4}) = 0. \tag{65}$$

The first row in Table 7 shows that the usual quadratic formula (2) with various computing systems leads to significantly different practical conclusions.

- With the 10-digit HP-15C, $\tilde{z}_+ = 0$: *none* of the acid dissociates.
- With the 12-digit HP 48GX, $\tilde{z}_+ = 1$: *all* of the acid dissociates.

Table 7. Formulae (2), (4), (11), (7) for the dissolved fraction $\tilde{z}_+$ or $\tilde{z}_1$.

$10^{-15} * z^2 + 6.76 \cdot 10^{-4} * z - 6.76 \cdot 10^{-4} = 0$				
SYSTEM[a]	HP-15C	HP 48GX	`MATLAB`	*Mathematica*
(2)	0	1	1.0000$\underline{1387379201}$	1.0000$\underline{13873792005}$
(4)	1	1	0.99999999999852	0.9999999999985207
(11)	1	0.999999999999	0.99999999999852	0.9999999999985207
(7)	0.999$\underline{617156}$	0.99999$\underline{5279049}$	0.9999999999$\underline{6824}$	0.9999999999$\underline{68238}$

[a] *Mathematica 3.0* & `MATLAB 5.2` computations with a Power Macintosh G3.

- After conversion to binary arithmetic, $\tilde{z}_+ = 1.0000\underline{1387}\ldots > 1$: a spontaneous generation of *more* acid dissociates than was initially mixed.

The moral could be that the more you pay for a machine, the more you get out of it. In contrast, with (4), both calculators give the same result, $\tilde{z}_1 = 1$, suggesting that all of the acid dissociates. With (11), the 12-digit calculator suggests that all but one part per trillion dissociates. Exact verifications corroborate this last result to all 12 digits. To this end, define

$$Q(z) = 10^{-15} * z^2 + (6.76 * 10^{-4}) * z - (6.76 * 10^{-4}). \tag{66}$$

From (64) it follows that $z_1 * z_2 = -K/C < 0$, whence $z_2 < 0 < z_1$. From $Q(0) = -K < 0 < C = Q(1)$, it then follows that $0 < z_1 < 1$, whence integer computations with *Mathematica* give the lower bound

$$1 - \frac{10^{-11}}{6.76} = 1 - \frac{C}{K} < 1 - \frac{C * z_1^2}{K} = z_1 < 1, \tag{67}$$

$$z_{1,\min} = 0.999\,999\,999\,998\,520\,710\,059\,171 < z_1 < 1, \tag{68}$$

and hence substitution of $z_{1,\min}$ in equation (64) gives the upper bound

$$z_1 < 1 - \frac{C * z_{1,\min}^2}{K} < 0.999\,999\,999\,998\,520\,710\,059\,176. \tag{69}$$

Because $C \ll K$, some texts advise to compute the solution z_1 as $1 - C/K$ [**31**, p. 355] or by iterations [**1**, p. 58], [**32**, p. 78]. However, the threshhold between $\ll$ and $<$ depends on the computing system, and for this type of example the theory shows that algorithm 1 already achieves a comparable accuracy [**26**]. The theory also explains the larger errors observed in Table 7 with (4), where the rounding errors in $2 \circledast a$ and $2 \circledast c$ affect all of the final results, in contrast to (11), where the rounding error in $b \oslash 2$ affects only the discriminant, and hence only about one half of the final results, because $|a * c| \ll h^2$ and hence $d = \sqrt{d^2}$ is only about one half of $-t = h + d$ [**26**]. ■

Irrelevance of Data

The need for more digits in the computations than in the data had already been pointed out by W. Edwards Deming,

> To secure two figures [...], one must carry *a*, *b*, and *c* through the fourth decimal; this is so in spite of the fact that we can not possibly rely statistically on so many figures in *a*, *b*, and *c* [...]—W. E. Deming [**5**, p. 236],

and by James Hardy Wilkinson,

> Practical computer users sometimes find it difficult to believe that there can be any justification for determining the coefficients of an explicit polynomial to high accuracy when the primary data are known to much lower accuracy. The fallacy in this argument may be exposed by quite a simple example.—J. H. Wilkinson [**39**, p. 14].

Example 11. For the chemical equation (65) in example 10, the inequalities $1 - (C/K) < z_1 < 1$ show that the computed solution $\tilde{z}_1 = 1$ is accurate to 11 significant digits for all data such that $C/K < 5 \cdot 10^{-12}$, regardless of whether the datum $K = 6.76 \cdot 10^{-4}$ has any correct significant digit. ■

In other words, in example 10 algorithm 1 yields solutions accurate to their last digit regardless of the number of accurate digits in the data.

In contrast, the first entry of Table 7 shows how the usual quadratic formulae (2) can fail to deliver any correct significant digit, and produce wrong orders of magnitude (0 instead of 1), with computations carrying more than three times as many digits (10) as the data contain (3).

Conclusions

1. The number of significant digits anywhere—in the data and in the computations—can be irrelevant.
2. The algorithm is the factor that determines the accuracy of the computed solutions.
3. Algorithm 1 delivers a greater accuracy than any of the other algorithms examined here.
4. The other algorithms can lead to significantly erroneous conclusions in practice.

Acknowledgment. This work was supported in part by a Faculty Research Grant from Eastern Washington University, and completed in part at the University of Washington.

References

1. Forman S. Acton, *Numerical Methods that Work*, Mathematical Association of America, 1990.
2. H. T. R. Aude, The solutions of the quadratic equation obtained by the aid of trigonometry, *National Mathematics Magazine* **13**(3):118–121, December 1938.
3. S. D. Conte and Carl de Boor, *Elementary Numerical Analysis*, McGraw-Hill, 2nd ed., 1972.
4. R. M. Corless, Six, lies, and calculators, *The American Mathematical Monthly* **100**(4):344–350, April 1993.
5. W. Edwards Deming, *Statistical Adjustment of Data*, Dover, 1964.
6. Gulio Carlo di Fagnano, *Produzioni mathematiche del Conte Gulio Carlo di Fagnano, Marchese de' Toschi, e di Sant' Onorio nobile romano, e patrizio senogagliese alla s antit a' din. s. Benedetto XIV, Pontefice Massimo, Tomo Primo*, Gavelli, Pesaro, Italy, 1750.
7. Leonard Eugene Dickson, *Elementary Theory of Equations*, Wiley, 1914.
8. F. C. Duncan and D. H. R. Huxtable, The DEUCE alphacode translator, *The Computer Journal* **3**(2):98–107, 1960.
9. Alan Edelman and H. Murakami, Polynomial roots from companion matrix eigenvalues, *Mathematics of Computation* **64**(210):763–776, April 1995.
10. Neil Eklund, Cordic: Elementary function computation using recursive sequences, *College Mathematics Journal* **32**(5):330–333, November 2001.
11. George Elmer Forsythe, Pitfalls in computation, or why a math book isn't enough, *The American Mathematical Monthly* **77**(9):931–956, November 1970.
12. Arthur A. Frost and Ralph G. Pearson, *Kinetics and Mechanism: A Study of Homogeneous Chemical Reactions*, Wiley, 2nd ed., 1961.
13. David Goldberg, What every computer scientist should know about floating-point arithmetic, *ACM Computing Surveys* **23**(1):5–48, March 1991.
14. Günther Hämmerlin and Karl-Heinz Hoffmann, *Numerical Mathematics*, Springer-Verlag, 1991.
15. Richard Wesley Hamming, *Numerical Methods for Scientists and Engineers*, Dover, 2nd ed., 1986.
16. Peter Henrici, *Essentials of Numerical Analysis with Pocket Calculator Demonstrations*, Wiley, 1982.
17. Hewlett-Packard Co., Corvallis Division, 1000 NE Circle Blvd., Corvallis, OR 97330, USA, *HP-15C Advanced Functions Handbook*, June 1984.
18. Hewlett-Packard Co., Corvallis Division, 1000 NE Circle Blvd., Corvallis, OR 97330, USA, *HP 48G Series User's Guide*, 7th edition, March 1994.

19. Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, 1996.
20. F. B. Hildebrand, *Introduction to Numerical Analysis*, McGraw-Hill, 1956.
21. Eugene Isaacson and Herbert Bishop Keller, *Analysis of Numerical Methods*, Wiley, 1966.
22. Cornelius Lanczos, *Applied Analysis*, Dover, 1988.
23. The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760-1500, USA, *MATLAB: High-Performance Numeric Computation and Visualization Software, Reference Guide*, October 1992.
24. U. W. Kulisch & W. L. Miranker, The arithmetic of the digital computer: A new approach, *SIAM Review* **28**(1):1–40, March 1986.
25. David E. Muller, A method for solving algebraic equations using an Automatic computer, *Mathematical Tables and Other Aids To Computation* **10**(56):208–215, October 1956.
26. Yves Nievergelt, Accuracy of numerical solutions of complex quadratic equations, preprint.
27. C. W. Schelin, Calculator function approximation. *The American Mathematical Monthly* **90**(5):317–325, May 1983.
28. Scott R. Schmedel, A math footnote, *The Wall Street Journal* **109**(62):1, Wednesday, 28 September 1988.
29. Internal Revenue Service, A codification of documents of general applicability and future effect, in *Code of Federal Regulations*, volume 26, pages 45,454–45,462 and 200–203, National Archives and Records Administration, April 1990, Internal Revenue Service, Office of the Federal Register.
30. H. L. Smith, Note on the quadratic formula, *Mathematics News Letter* **3**(7):19–20, March 1929.
31. R. Nelson Smith and Conway Pierce, *Solving General Chemistry Problems*, W. H. Freeman, 5th ed., 1980.
32. Ralph G. Stanton, *Numerical Methods for Science and Engineering*, Prentice-Hall, 1961.
33. Pat H. Sterbenz, *Floating-Point Computation*, Prentice-Hall, 1974.
34. Josef Stoer and Roland Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, 2nd ed., 1993.
35. Kim-Chuan Toh and Lloyd N. Trefethen, Pseudozeros of polynomials and pseudospectra of companion matrices, *Numerische Mathematik* **68**(3):403–425, September 1994.
36. Richard D. C. Trainer, *The Arithmetic of Interest Rates*, Federal Reserve Bank of New York, Public Information Department, 33 Liberty Street, New York, NY 10045, 1982.
37. Bartel Leenert van der Waerden, *Geometry and Algebra in Ancient Civilizations*, Springer-Verlag, 1983.
38. James Hardy Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall Series in Automatic Computation, Prentice-Hall, 1963.
39. James Hardy Wilkinson, The perfidious polynomial, in Gene H. Golub, editor, *Studies in Numerical Analysis*, volume 24 of *MAA Studies in Mathematics*, pages 1–28, Mathematical Association of America, 1972.
40. James Hardy Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, paperback edition, 1988.
41. Stephen Wolfram, *The Mathematica Book*, Wolfram Media, 3rd ed., 1996.

Alcohol in Minnesota

Stan Lipovetsky (lipovetsky@customresearch.com) sends a copy of the *Minnesota Drivers Manual* (Minnesota Department of Public Safety, 2001) with page 85 dog-eared. On it we find

> **Reaction time**
> The depressant effect of alcohol reduces reaction time.

"So," asks Dr. Lipovetsky, "to drink or to think?" He calls it an interesting example of an inequality described in words.