QUADRATIC PROGRAMMING?

WILLIAM Y. SIT

Department of Mathematics,
The City College of The City University of New York,
New York, NY 10031, USA
E-mail: wyscc@cunyvm.cuny.edu

This is a talk on how to program a computer to solve the quadratic equation optimally. The optimal algorithm depends on understanding of the floating point system and its limitations.

1. Introduction

In this talk[a], we study the problem of solving a quadratic equation.

$$Ax^2 + Bx + C = 0 \tag{1.1}$$

where the coefficients A, B, C are real numbers.

This presentation is entirely based on an article by Sterbenz [1]. We recall the well-known formula for the solutions of (1.1) below.

Theorem 1.2. *The solutions (using possibly complex numbers) to Equation (1.1) is given by the quadratic formula*

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{1.3}$$

Furthermore, we have three cases:

(a) *If $B^2 - 4AC > 0$, we have two real distinct roots.*
(b) *If $B^2 - 4AC = 0$, we have a repeated real root.*
(c) *If $B^2 - 4AC < 0$, we have two conjugate complex roots.*

Proof. This is proved by completing the square. □

While the formula is simple, the computation to approximate the exact solutions using floating point numbers is far from trivial.

[a]Presented to CCNY Math Club on November 20, 1997

2. Floating Point Systems

A floating point system is used in the hardware arithmetic logic unit of a computer. A floating point system consists of a finite set of rational numbers used to represent the entire real line. This of course necessarily means that entire intervals on the real line are mapped into each rational number in the system. To understand numerical computations, we must understand how this inadequate representation affects computations.

Floating point systems are usually described by means of how the rational numbers are stored in memory or registers. Typically, 32 bits are used (single precision). An example of how the bits may be allocated is the following:

bit(s)	Usage	Range
31	sign of number	$\pm$
24–31	exponent e	-64 to +63
0–23	mantissa (normalised) m	$0.1 \le m < 1$

Thus all the positive rational numbers are of the form $x = mr^e$ with $0.1 \le m < 1$ and r is the radix. The radix 16 is often used. An important consequence of the finiteness of a floating point system is that there is a smallest positive number $\omega = 16^{-65}$ and a largest positive number $\Omega \approx 16^{63}$. If at any time, a computation produces a value whose absolute value exceeds Ω, an *overflow* occurs. Similarly, if the absolute value is less than ω, an *underflow* occurs. Exactly how these unpleasant situations are handled is system dependent. In this talk, we shall view these as error conditions.

3. Goal

Our goal is to compute the roots of the quadratic equation as accurately as the floating point system used allows, without encountering underflow or overflow along the way unless the roots themselves have absolute values either $< \omega$ or $> \Omega$.

Assumption 3.1.

The following assumptions are used:

(a) *All coefficients and arithmetic operations are performed in a floating point system* $FP(r, p, e_1, e_2, c)$, *where* r *is the radix (taken as 16),* p *is the number of digits in the mantissa, the exponent* e *satisfies* $e_1 \le e \le e_2$ (*say* $e_1 = -64, e_2 = 63$), *and* c *denotes that all rounding is done by chopping.*

(*b*) *The coefficients A, B, C are given as normalized floating point numbers.* (*This avoids problem caused by base conversion.*)
(*c*) *We assume that $A \neq 0$.*
(*d*) *The relative error introduced by the square root function is small.*
(*e*) *Multiplication or division will have small relative error if the operands do.*
(*f*) *Addition or subtraction will have small relative error unless we are in the subtract magnitude case.*
(*g*) *We assume we can extract the parts of a floating point number efficiently.* (*Have you ever tried doing this in a high level language? Note that these routines are machine dependent!*)

4. Detailed Analysis of Computation

We shall divide the computation based on Equation (1.2) in several stages.

Computing B^2 and $4AC$

We can encounter overflow even when the coefficients and roots are substantially less than Ω.

Example 4.1. Consider the equation:

$$16^{40}x^2 - 3 \cdot 16^{40}x + 2 \cdot 16^{40} = 0. \tag{4.2}$$

Both B^2 and $4AC$ overflow, but the roots are 1 and 2.

Can we scale the equation first so that the coefficient of x^2 is close to 1? In Example 4.1, we can do this by multiplying the equation by 16^{-40}.

Example 4.3. Consider the equation:

$$16^{-40}x^2 - 3x + 2 \cdot 16^{40} = 0. \tag{4.4}$$

Scaling will cause an overflow on the constant term. The roots are 16^{40} and $2 \cdot 16^{40}$.

To solve this problem, we have to use a different type of scaling: namely, substitute $x = 16^{40}y$. This then leads to Equation (4.2) of Example 4.1 with y replacing x.

In general, we have to apply both types of scaling. To avoid errors during scaling, we use only scaling factors that are powers of r. Suppose we substitute $x = 16^K y$ into Equation (1.1) and then multiply the coefficients by 16^L to obtain:

$$16^{2K+L}Ay^2 + 16^{K+L}By + 16^L C = 0, \tag{4.5}$$

which we rewrite as $A'y^2 + B'y + C' = 0$. We shall choose K and L so that

- the exponent $e_{A'}$ of A' is 0.
- either $B = 0$ or the exponent $e_{B'}$ of B' is 0.

Let e_A, e_B, e_C respectively be the exponents of A, B, C.

Case: $B = 0$. Set $K = 0, L = -e_A$.

Case: $B \neq 0$. The exponents satisfy the equations:

$$\begin{aligned} 0 = e_{A'} &= 2K + L + e_A \\ 0 = e_{B'} &= K + L + e_B \\ e_{C'} &= L + e_C \end{aligned}$$

Solving the first two equations for K and L gives

$$K = e_B - e_A, \quad L = e_A - 2e_B.$$

So we can obtain A' and B' by changing the exponents of A and B to zero!

Question 4.6. Can we change the exponent of C using $e_{C'} = e_C + L$?

Example 4.7. Consider the equation

$$16^{-40}x^2 + 16^{-40}x - 16^{40} = 0. \tag{4.8}$$

Problem. Even if $|C'| < \Omega$, the quantity $4A'C'$ may still overflow! We have

$$\begin{aligned} 16^{-1} &\le |A'| < 1 \\ 16^{-1} &\le m_{C'} < 1 \\ -64 &\le e_{C'} \le 63 \end{aligned}$$

and hence

$$\omega = 16^{-65} \le 4 \cdot 16^{e_{C'}-2} \le 4|A'C'| < 4 \cdot 16^{e_{C'}} < \Omega \approx 16^{63}$$

if and only if

$$-63 \le e_{C'} \le 62. \tag{4.9}$$

First suppose Inequality (4.9) does not hold.

Case: $B = 0$. Write $C' = 16^{2M} C''$, where the exponent of C'' is either 0 or 1. Clearly, C'' can be obtained by changing the exponent of C' and hence of C. Since $B = 0$, the roots y are given by $\pm\sqrt{C'/A'} = \pm 16^M \sqrt{C''/A'}$. Compute

$$S = \sqrt{\left|\frac{C''}{A'}\right|}.$$

If $AC < 0$, the roots of Equation (1.1) are obtained by adding $K + M$ to the exponents of S and $-S$. Similarly, if $AC > 0$, the roots are $\pm 16^{K+M} Si$, where $i = \sqrt{-1}$.

Case: $B \ne 0$. If $e_{C'} < -63$, we have

$$16^{-1} \le |B'| < 1, \quad |4A'C'| < 4 \cdot 16^{-63} < 16^{-62}.$$

Thus we can safely ignore the term $4|A'C'|$ in computing $B'^2 - 4A'C'$.

If $e_{C'} > 62$, Then we may ignore the term B'^2. If $AC < 0$, the roots are real, and we may ignore the term B' as well. This reduces to the case $B = 0$ above. If $AC > 0$, then the roots are complex, with the real parts of y given by $-B'/(2A')$ and imaginary parts $\pm 16^M S$ as in the case above.

In the remainder of the talk, we assume that Inequality (4.9) holds, that we can change the exponent of C to $e_{C'}$, and that we have computed B'^2 and $4A'C'$.

Computing $B^2 - 4AC$

Note that if $\Delta' = B'^2 - 4A'C'$ and $\Delta = B^2 - 4AC$, then $\Delta' = 16^{2K+2L}\Delta = 16^{-2e_B}\Delta$. To simplify notation, we shall drop the prime's.

The subtraction of $4AC$ from B^2 can cause large relative errors if $B^2 \approx 4AC$. This is the subtract magnitude case.

Example 4.10. Suppose $B = 1^-$ and $\Delta \approx 16^{-6}$. If we use 6 digit hexadecimal arithmetic (for instance, $FP(16, 6, -64, 63, c)$), we may get $\Delta \approx 2{\cdot}16^{-6}$ (an error of 1 unit in the last digit) and $\sqrt{\Delta} = \sqrt{2} \times 16^{-3}$ instead of 16^{-3}. Thus the error in the roots will be in the middle of the digits.

A more general analysis of this situation is the following. When $B^2 \approx 4AC$, the roots are nearly equal. Consider the equation,

$$x^2 - 2ax + a^2 = 0.$$

If we change the two roots to $a \pm \epsilon$, the equation above becomes

$$x^2 - 2ax + a^2 - \epsilon^2 = 0,$$

that is, the coefficient C changes by a much smaller amount. Reversing the argument, a small change in C causes a much larger change in the exact roots. This is an ill-conditioned problem. It can be shown that the computed value for $\sqrt{\Delta}$ is exactly equal to $\sqrt{B^2 - 4A\overline{C}}$ for some $\overline{C} \approx C$.

The solution is to compute Δ in double precision (using 64 bits) and then truncate to single precision before computing the square root.

The Final Steps

Assuming that $\Delta > 0$, the penultimate step in computing the roots is to add $\pm\sqrt{\Delta}$ to $-B$. If we are not careful, one of the roots may be computed with large relative errors due to, again, the subtract magnitude case. Fortunately, by computing the right root first, we can avoid the bad subtraction. We should test the sign of $-B$ and select the sign of $\sqrt{\Delta}$ to agree with the sign of $-B$ so that we are in the add magnitude case. Having computed this root r_1, we can compute r_2 using the fact that the product $r_1 r_2$ is C/A.

The complex case when $\Delta < 0$ is actually simpler since the real and imaginary parts can be computed separately.

5. Ready for an Average Homework?

Exercise 5.1. The challenge is to compute the average of two normalized floating point numbers A and B. Requirements:

(a) The average must be a normalized floating point number C'.
(b) The value C' must be approximately $C = (A + B)/2$, and must have the same sign.
(c) Provide a reasonable bound for the error (either in terms of units in the last digit, or as a relative error).
(d) If we denote C' as $f(A, B)$, then f must satisfy the following:
   (i) $\min(A, B) \le f(A, B) \le \max(A, B)$
   (ii) $f(A, B) = f(B, A)$

(iii) $f(A, B) = 0$ if $B = -A$, unless C underflows.
(iv) $f(-A, -B) = -f(A, B)$

(e) The computation should never produce an overflow, and it should not produce an underflow unless $0 < |C| < \omega$.

Hints. Here are three formulae you might consider using (all operations denote floating point arithmetic operations):

Formula 1: $f_1(A, B) = (A + B)/2$
Formula 2: $f_2(A, B) = (A/2) + (B/2)$
Formula 3: $f_3(A, B) = A + ((B - A)/2)$

6. Conclusion

The purpose of this talk, depending on your point of view, is to show you either how difficult it is to program a quadratic equation solver on a programmable calculator, or to show how futile it is to use a calculator or even a programming language like FORTRAN. The moral is that it is time to learn to use computer algebra systems.

References

1. Pat H. Sterbenz, Floating-Point Computation, Prentice Hall, 1974.

Appendix I

This little "paper" is used to illustrate the changes made the style file `ws-procs9x6.cls` to obtain the ASCM proceedings style file `ascm9x6.cls` using LaTeX_2e. The commands that are illustrated in this file are:

(1) Changes to the count of theorem environments and equation environments. These are now all combined and counted in one sequence per section, and has the form $x.y$ where x is the current section number and y is the sequence number.
(2) The macro \paren, to typeset all parenthesis in upright font (non-italicised or slanted) in any environment. Examples in this paper occur in part (a) of Assumption 3.1 (note the use of manual parenthesis in the reference to part (a)), and implicitly in the \begin{alphlist} ... \end{alphlist} environment in Theorem 1.2.

(3) The use of nested `alphlist` environment. Note that this environment has *not* been changed except for the parenthesis. It is kind of counter-intuitive as provided by WSPC (see Exercise 5.1 above, where the nested environment is also `alphlist`). The WSPC style file limits all nesting to two levels and provides automatic choice for the inner list (in the case of `alphlist`, the inner is typeset using lower case roman numerals).
(4) Miscellaneous examples of theorem environments such as `assumption`, `question`, `exercise`.
(5) Use of un-numbered sections such as this Appendix.
(6) Use of citation in [1] style.
(7) Use of labels for section, theorem, parts, references.

Appendix II

This second Appendix is added in the "camera-ready" version. The changes are, after consultation with World Scientific:

(1) The use of unnumbered subsections, merging sections 4, 5, 6 into one. If the subsections had been numbered, they would have been 4.1, 4.2, 4.3 (which duplicates the theorem environment numbering). The editors discourage the use of subsections.
(2) Note that added "fillers" at the beginning of Section 4 and the use of label for the reference to the section. In general, it is editorially unpleasing to start a subsection immediately after a section heading without fillers.
(3) The use of \begin{description} environment (added by World Scientific to the ascm9x6 style file) in the Hints of Section 5 instead of the \begin{eqnarray*} environment. Note that this list environment uses arabic numerals in square brackets like [1], [2], by default, which unfortunately coincides with ASCM citation style. In this example, the numerals are replaced by definition style words and note the extension into the left margin of the item label and the manual boldface. The editors of the proceedings discourage the use of this environment.