How to Solve a Quadratic Equation

James F. Blinn

Microsoft Research

One of my favorite experiences in high school mathematics was learning how to solve quadratic equations. We all remember how it goes—we want to find values of x that satisfy the equation

$$ax^2 + bx + c = 0$$

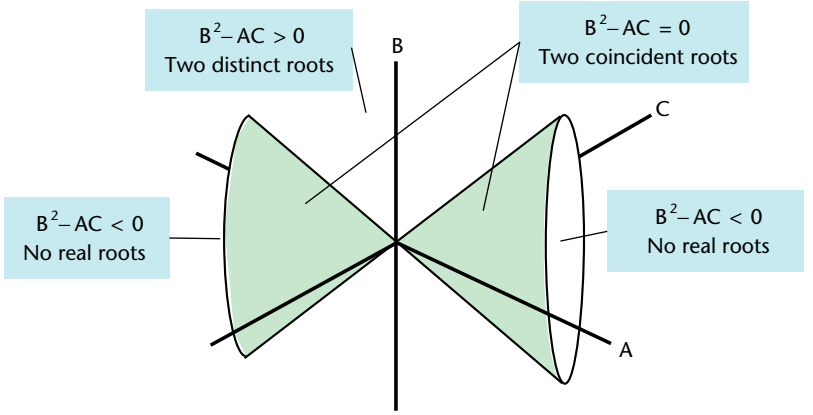
After some algebraic magic we find the solution for x as the quadratic formula. Let's all recite it together:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

I think it was the sheer nonobviousness of this solution that fascinated me so much.

Little did I know in high school that I would later use this equation extensively to make pictures of planets and DNA molecules. Each pixel in such an image requires the intersection of a second order surface with a ray connecting the eye and the pixel; so it's the solution of a quadratic equation. I've described some of these adventures in a series of articles titled "How to Draw a Sphere."[1]

As is usual with life, things are not as simple as they were in high school. There are several problems with the quadratic formula as written in Equation 1. In this column, I'll talk about these problems and various solutions to them.



1 The space of possible quadratics.

Neater coefficients

Before doing anything else, I'm going to change the representation of the inputs to the problem. It will turn out that later calculations will be much neater if we express the quadratic as

$$Ax^2 + 2Bx + C = 0$$

I've basically just pulled a factor of 2 out of the coefficient b and changed a, b, and c to uppercase. I like to use uppercase for coefficients as it helps to distinguish between inputs to the algorithm (uppercase) from intermediate results and outputs (lowercase). Mathematics is all about finding patterns, and anything we can do to make patterns more apparent visually, even within algebraic formulas, is good.

The standard quadratic formula then looks like

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

Figure 1 illustrates the universe of possible quadratics in ABC space. The surface where the discriminant $B^2 - AC$ is zero is a cone with its axis along the $A = C$ line, and with the A and C axes embedded in it. Note that it is not a circular cone; the cross section is elliptical. Points on this cone correspond to quadratics with double roots.

Inside the cone there are no real roots, outside the cone there are two distinct roots. Looking at the relative volumes involved, it's comforting to know that, statistically at least, you are much more likely to stumble upon a quadratic with two real roots as one with complex roots. In this article I'll only talk about real solutions, so I'll consider only polynomials where $B^2 - AC \geq 0$.

Numerical problems

The first problem with Equation 2 is numerical. A classic source of numerical problems is the subtraction of two nearly equal quantities. The higher order bits cancel out leaving a result that has only a few bits of accuracy. This will happen with Equation 2 whenever $B^2 >> AC$ so that

$$\sqrt{B^2 - AC} \approx |B|$$

 Published by the IEEE Computer Society

When calculating the roots according to Equation 2, one or the other of the expressions

$$-B-\sqrt{B^2-AC}$$

and

$$-B+\sqrt{B^2-AC}$$

results in subtraction of two nearly equal values. That calculated root can be mostly noise.

Fixing the numerical problem

The typical way to solve the numerical problem, as discussed in more detail elsewhere,[2] is to first note that the product of the two roots is

$$x_1x_2=\left(\frac{-B-\sqrt{B^2-AC}}{A}\right)\left(\frac{-B+\sqrt{B^2-AC}}{A}\right)=\frac{C}{A}$$

Because we know that finding one of the roots is numerically safe, we just calculate the other root as $x_2 = C/Ax_1$. We can find the safe root by looking at the sign of B and picking the solution that has us adding values instead of subtracting them. Press et al.[2] recommend an algorithm that is, in our notation,

$$\begin{aligned} q&=-\left(B+\text{sgn}\left(B\right)\sqrt{B^2-AC}\right)\\ x_1&=q/A\\ x_2&=C/q \end{aligned} \tag{3}$$

Does this really matter?

It's worthwhile understanding exactly when this numerical problem happens. The condition

$$\sqrt{B^2-AC}\approx\left|B\right|$$

means that one of the two roots is much smaller (in magnitude) than the other, a perfectly likely occurrence. For example, consider the polynomial

$$f(x)=x^2+2\times1000000x+1=0 \tag{4}$$

For comparison purposes, let's calculate the roots in extremely high precision. The basic interesting quantity is

$$\begin{aligned}\sqrt{B^2-AC}&=\sqrt{1000000^2-1}\\&=\sqrt{999999999999}\\&=999999.999999499999999999875\ldots\end{aligned}$$

According to Equation 2 the roots are, then

$$\begin{aligned} x_1&=-1000000+999999.999999499999999999875\ldots\\ &=-0.000000500000000000124999999999996850283\ldots\\ x_2&=-1000000-999999.999999499999999999875\\ &=-1999999.999999499999999999875 \end{aligned} \tag{5}$$

However, when we do this with single-precision floating point we get the approximate intermediate results:

$$\begin{aligned}B^2-AC&\cong999999995904\\\sqrt{B^2-AC}&\cong1000000\end{aligned}$$

If we were to blindly use Equation 2 with these approximate values we would get

$$\begin{aligned}x_1&=-1000000+1000000=0\\x_2&=-1000000-1000000=-2000000\end{aligned}$$

On the other hand, if we use Equation 3 we would get

$$x_1=1/x_2=-0.0000005$$

This, of course, is a better match to the result in Equation 5. But this might not always be all that big a deal. For example, consider the accuracy with which we know x_2. For single-precision floating-point numbers, the representable values surrounding this root are

–1999999.875
–2000000.000
–2000000.125

That is, the least significant bit for a number of magnitude 2000000 has the value 0.125. This means that the accuracy with which we can represent the larger root, simply due to floating-point representation limitations, is much bigger than the error we got in the smaller root by using Equation 2. Of course, the relative error in the smaller root is much better using Equation 3. But if, for example, the x value represents a position in space, knowing one root to high relative precision might not be all that useful if the absolute error of the other root is large. Or consider the following. Let's evaluate the function $f(x)$ from Equation 4 at the various numerically calculated roots and see how close to zero we get:

$$\begin{aligned}f\left(-2000000\right)&=1\\f\left(-.0000005\right)&=.0000005^2\\f\left(0\right)&=1\end{aligned}$$

Sure, using the value –.0000005 for x_1 gives us something really small. But using the value zero for x_1 gives us a value of f that is no worse than that for the best approximation we can get to the other root x_2.

All in all though, I recommend Equation 3 (with some modifications that we'll discuss next). But in some situations it might be useful to consider whether it's really necessary.

Root ordering

The formulation in Equation 3 works numerically but has a few problems for computer animation (which is notoriously hard on algorithms). First, the algorithm as stated returns values that are discontinuous in B. If we smoothly animate B from positive to negative (a possible scenario), the x_1, x_2 pair swaps places from being the (lowest, highest) roots to being the (highest, lowest)

roots (or vice versa, depending on the sign of A). This could create various kinds of havoc in rendering algorithms. For this reason I prefer the following variant:

$$\begin{aligned}&\text{if}\left(B<0\right)\\&\qquad q=-B+\sqrt{B^2-AC}\\&\qquad x_1=q/A\\&\qquad x_2=C/q\\&\text{else}\\&\qquad q=+B+\sqrt{B^2+AC}\\&\qquad x_1=-C/q\\&\qquad x_2=-q/A\end{aligned}\tag{6}$$

This guarantees that $x_2 \le x_1$ (if $A > 0$) and that $x_1 \le x_2$ (if $A < 0$) independently of what happens to B and C. If you are solving a lot of quadratics it might be possible (and worthwhile) to have some global constraint that guarantees that $A > 0$ (note that we can always flip the sign of all three of A, B, C and get a quadratic with the same roots). You might also be tempted to simply divide A out of B and C as a preprocess and perform some optimizations using the fact that now effectively $A = 1$. I'm not going to do that here since I will shortly enter into homogeneous land where we can have $A = 0$.

Again there are some tradeoffs. Equation 3 is more likely to parallelize nicely because it doesn't have conditionals. (The sgn function just requires bit shuffling.) So if you don't care about the ordering of the returned roots, Equation 3 might seem to be a better choice. But first consider the following.

A degeneracy

Both Equations 3 and 6 have an Achilles heel that Equation 2 does not. If it happens that $B = C = 0$ (a possible occurrence), Equations 3 and 6 go up in smoke. One of the solutions will come out as 0/0 even though the quadratic has a perfectly respectable double root at 0. Since we are testing the sign of B anyway, we can head off this problem by including a case for $B = 0$. If $B = 0$ we don't have the numerical problem so we can use Equation 2, which gives us

$$x=\pm\frac{\sqrt{-AC}}{A}$$

Let us, then, fix up the algorithm to include this test. I'll present the algorithm as Table 1, with the three conditions along the three rows and the results down the columns. Note that I have made some of the cells in this table white. This is to call attention to the pattern that for $B = 0$ the calculation for x_1 is the same as that for $B < 0$ and the calculation for x_2 is the same as that for $B > 0$.

Table 1. Algorithm.

Conditions	q	x_1	x_2
B > 0	$+B+\sqrt{B^2-AC}$	$-\frac{C}{q}$	$-\frac{q}{A}$
B = 0	$+\sqrt{-AC}$	$\frac{q}{A}$	$-\frac{q}{A}$
B < 0	$-B+\sqrt{B^2-AC}$	$\frac{q}{A}$	$\frac{C}{q}$

Going homogeneous

Now what happens if $A = 0$? Conventionally, this means that the quadratic degenerates into a linear equation:

$$2Bx + C = 0$$

But we're not going to think that way. Instead, I want to generalize our problem slightly into that of solving a homogeneous quadratic equation:

$$Ax^2 + 2Bxw + Cw^2 = 0$$

This allows us to write the equation as a matrix product, and writing things as matrices is always good:

$$Ax^2+2Bxw+Cw^2=\begin{bmatrix}x & w\end{bmatrix}\begin{bmatrix}A & B\\B & C\end{bmatrix}\begin{bmatrix}x\\w\end{bmatrix}=0\tag{7}$$

For example, the discriminant of the quadratic can now be seen as (minus) the determinant of the matrix:

$$B^2-AC=-\det\begin{bmatrix}A & B\\B & C\end{bmatrix}$$

Now we are looking for $[x, w]$ pairs that satisfy this equation. And any nonzero multiple of such an $[x, w]$ pair also satisfies it. Computer graphicists familiar with 3D homogeneous coordinates can think of $[x, w]$ as a 1D homogenous coordinate location on the number line. You get the actual solution by the homogeneous division x/w. Going homogeneous basically adds a value at infinity to the number line at $[x, w] = [1, 0]$.

Now we can see that the condition $A = 0$ simply means that the homogeneous quadratic has one root at infinity:

$$2Bxw + Cw^2 = (2Bx + Cw)w = 0$$

And the two roots are any scalar multiples of

$$\begin{bmatrix}x_1,w_1\end{bmatrix}=\begin{bmatrix}C,-2B\end{bmatrix}$$
$$\begin{bmatrix}x_2,w_2\end{bmatrix}=\begin{bmatrix}1,0\end{bmatrix}$$

We just need to beef up our algorithm to generate homogeneous results (easy) and to work properly for any values of (A, B, C), in particular for any combinations of A, B, C being zero (slightly trickier). In addition, we expect that the final algorithm will be nicely symmetrical with respect to A and C, in the same manner as Equation 7.

With a homogeneous quadratic, the solutions we require are now $[x, w]$ pairs, whose ratio equals the nonhomogeneous solution we got previously. The nonhomogeneous solutions are already ratios, so recasting them in this 1D homogeneous coordinate notation is easy (see Table 2).

This version works fine for all cases except one, where $A = B = 0$. This case is symmetrical to the one we've already covered, where $C = B = 0$. This latter case correctly generates a double root at [0, 1]. We would like for the $A = B = 0$ case to generate a double root at infinity, [1, 0]. Instead, it generates [0, 0]. To fix this let's look more closely at our choices for $[x_1, w_1]$. I'll now give names to the two homogeneously equivalent candidates:

$$\mathbf{p} = \begin{bmatrix} -C & +B + \sqrt{B^2 - AC} \end{bmatrix}$$
$$\mathbf{m} = \begin{bmatrix} -B + \sqrt{B^2 - AC} & A \end{bmatrix}$$

If $B > 0$ we pick $\mathbf{p}$, and if $B < 0$ we pick $\mathbf{m}$. If $B = 0$ these choices simplify to

$$\mathbf{p}_0 = \begin{bmatrix} -C & +\sqrt{-AC} \end{bmatrix}$$
$$\mathbf{m}_0 = \begin{bmatrix} \sqrt{-AC} & A \end{bmatrix}$$

Usually either one of these will work; they are homogeneous scales of each other. But if, in addition, we have $C = 0$ we do not want to pick $\mathbf{p}_0$; we have to pick $\mathbf{m}_0$. Contrariwise, if $A = 0$ we do not want to pick $\mathbf{m}_0$; we have to pick $\mathbf{p}_0$. We can avoid testing floating-point numbers against zero (always a dicey proposition) by instead seeing which of A and C are largest in magnitude and picking appropriately:

$$\begin{bmatrix} x_1, w_1 \end{bmatrix} = \text{if } \left(|A| > |C| \right) \text{ then } \mathbf{m}_0 \text{ else } \mathbf{p}_0$$

Similarly, for $[x_2, w_2]$ we have the two choices:

$$\mathbf{q} = \begin{bmatrix} -B - \sqrt{B^2 - AC} & A \end{bmatrix}$$
$$\mathbf{n} = \begin{bmatrix} C & -B + \sqrt{B^2 - AC} \end{bmatrix}$$

When $B = 0$, these simplify to

$$\mathbf{q}_0 = \begin{bmatrix} -\sqrt{-AC} & A \end{bmatrix}$$
$$\mathbf{n}_0 = \begin{bmatrix} C & +\sqrt{-AC} \end{bmatrix}$$

And the choice becomes

$$\begin{bmatrix} x_2, w_2 \end{bmatrix} = \text{if } \left(|A| > |C| \right) \text{ then } \mathbf{q}_0 \text{ else } \mathbf{n}_0$$

Putting it together

Table 3 shows the final homogeneous algorithm. In this version I've shown it using the vector names instead of their values to emphasize the pattern in the entries. Recall that we were careful to make the $[x_1, w_1]$ solution stay stable when B goes from positive to zero to negative. That is, we made sure that the $[x_1, w_1]$ solution doesn't abruptly jump over to the other solution during this transition. There is, however, a jump via a homogeneous scale when B passes zero. Finally, note that if $B = 0$ and $|A| = |C|$, there doesn't seem to be a good reason for picking one solution set over another. I've arbitrarily chosen the $\mathbf{m}_0$ and $\mathbf{q}_0$ pair.

Table 2. Incomplete homogeneous algorithm.

Condition	q	$[x_1, w_1]$	$[x_2, w_2]$
$B > 0$	$+B + \sqrt{B^2 - AC}$	$[-C, q]$	$[-q, A]$
$B = 0$	$+\sqrt{-AC}$	$[q, A]$	$[-q, A]$
$B < 0$	$-B + \sqrt{B^2 - AC}$	$[q, A]$	$[C, q]$

Table 3. Final homogeneous algorithm.

Condition		$[x_1, w_1]$	$[x_2, w_2]$
$B > 0$		$\mathbf{p}$	$\mathbf{q}$
$B = 0$	$\lvert A \rvert \geq \lvert C \rvert$	$\mathbf{m}_0$	$\mathbf{q}_0$
	$\lvert A \rvert < \lvert C \rvert$	$\mathbf{p}_0$	$\mathbf{n}_0$
$B < 0$		$\mathbf{m}$	$\mathbf{n}$

We're not done yet

There is another way to look at this. Numerical considerations aside, we have found two possible formulations for each of the two homogeneous roots:

$$\begin{bmatrix} x_1 & w_1 \end{bmatrix} = \begin{bmatrix} -B - \sqrt{B^2 - AC} & A \end{bmatrix} \text{ or } \begin{bmatrix} C & -B + \sqrt{B^2 - AC} \end{bmatrix}$$
$$\begin{bmatrix} x_2 & w_2 \end{bmatrix} = \begin{bmatrix} -B + \sqrt{B^2 - AC} & A \end{bmatrix} \text{ or } \begin{bmatrix} C & -B - \sqrt{B^2 - AC} \end{bmatrix}$$

The two formulations for each of $[x_1, w_1]$ and $[x_2, w_2]$ are homogeneously equivalent; one is a scalar multiple of the other. But what does the existence of two-solution formulations really mean? Next time, we'll see that these are special cases of a more general solution scheme, and also find an intriguing relationship between the transformation of $[x, w]$ parameter space and corresponding transformations in $[A, B, C]$ coefficient space. We'll use this to come to a deeper understanding of what we are really doing when we solve quadratic equations. ■

References

1. J.F. Blinn, *Notation, Notation, Notation*, Morgan Kaufmann, 2003.
2. W.H. Press et al., *Numerical Recipes in C*, Cambridge Univ. Press, 1988, p. 184.

Readers may contact Jim Blinn at blinn@microsoft.com.