Low-Degree Polynomial Roots

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/

Created: July 15, 1999
Last Modified: November 3, 2012

Contents

1 Quadratic Roots **3**

1.1 Two Simple Real-Valued Roots 4

1.2 One Double Real-Valued Root 4

1.3 Two Simple Complex-Valued Roots 4

1.4 Numerical Issues 4

1.4.1 Discriminant Nearly Zero 4

1.4.2 Leading Coefficient Nearly Zero 5

1.4.3 Cancellation of Significant Digits 5

1.4.4 Root Deflation 10

2 Cubic Roots **10**

2.1 Constant Term is Zero 11

2.1.1 One Triple Real-Valued Root 11

2.1.2 Three Simple Real-Valued Roots 11

2.1.3 One Simple Real-Valued Root, Two Simple Complex-Valued Roots 11

2.2 Constant Term is Not Zero 11

2.2.1 One Simple Real-Valued Root, One Double Real-Valued Root 12

2.2.2 One Simple Real-Valued Root, Two Simple Complex-Valued Roots 13

2.2.3 Three Simple Real-Valued Roots 14

2.3 Summary of Real-Valued Root Construction 14

2.4 Numerical Issues 15

3 Quartic Roots **15**

3.1 Special Cases 16

3.1.1 Constant Coefficient is Zero 16

3.1.2 Linear Coefficient is Zero 16

3.1.3 Quadratic Coefficient is Zero 16

3.2 General Cases 17

3.2.1 Nonsimple Roots 17

3.2.2 Real-Valued Root of Multiplicity 3 18

3.2.3 Real-Valued Root of Multiplicity 2 18

3.2.4 The General Construction 19

4 Real Parts of Polynomial Roots **19**

The roots of polynomials of degrees 2, 3, or 4 can be found using closed-form algebraic expressions. This document presents the constructions of those expressions and assumes (1) the coefficients are real-valued and (2) exact arithmetic (a theoretical construction). Some discussion is also provided for numerical issues when solving for real-valued roots using floating-point arithmetic.

If a root occurs once in the list of roots, it is referred to as a *simple root*, wwhether real-valued or complex-valued. A root occurring two times, three times, or four times is referred to as a *double root*, *triple root*, or *quadruple root*, respectively. The number of times a root occurs in the list is called its *multiplicity*. The multiplicities are 1, 2, 3, or 4 for simple, double, triple, or quadruple roots, respectively.

For real-valued coefficients, complex-valued roots occur in pairs as a root $z = u+v\boldsymbol{\imath}$ and its complex conjugate $\bar{z} = u - v\boldsymbol{\imath}$, where $\boldsymbol{\imath}^2 = -1$.

1 Quadratic Roots

The general quadratic equation is of the form

$$f(x) = a_2x^2 + a_1x + a_0 = 0 \tag{1}$$

where $a_2 \neq 0$. The quadratic polynomial is said to be *monic* when the second-degree coefficient is one,

$$f(x) = x^2 + b_1x + b_0 = 0 \tag{2}$$

The monic quadratic equation is obtained from the general equation by dividing by a_2, in which case $b_1 = a_1/a_2$ and $b_0 = a_0/a_2$.

The roots to Equation (2) are obtained by *completing the square*. Specifically, the x^2 and x terms are made part of the square of a linear expression,

$$0 = x^2 + b_1x + b_0 = \left(x^2 + b_1x + \frac{b_1^2}{4}\right) + b_0 - \frac{b_1^2}{4} = \left(x + \frac{b_1}{2}\right)^2 + \frac{4b_0 - b_1^2}{4} \tag{3}$$

Subtracting leads to

$$\left(x + \frac{b_1}{2}\right)^2 = \frac{b_1^2 - 4b_0}{4} \tag{4}$$

Taking the square root, subtracting a term, and combining terms leads to

$$x = \frac{-b_1 \pm \sqrt{b_1^2 - 4b_0}}{2} \tag{5}$$

The *discriminant* is $\Delta = b_1^2 - 4b_0$. Its name refers to its use in classifying the roots of the equation. Substituting $b_1 = a_1/a_2$ and $b_0 = a_0/a_2$ leads to roots for Equation (1),

$$x = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0a_2}}{2a_2} \tag{6}$$

This equation has discriminant $\hat{\Delta} = a_1^2 - 4a_0a_2$.

1.1 Two Simple Real-Valued Roots

If $\Delta > 0$, the two roots are real-valued and distinct, $r_0 = (-b_1 - \sqrt{\Delta})/2$ and $r_1 = (-b_1 + \sqrt{\Delta})/2$.

1.2 One Double Real-Valued Root

If $\Delta = 0$, the equation has a single real-valued root, $r_0 = -b_1/2$, which is a double root (multiplicity 2).

1.3 Two Simple Complex-Valued Roots

If $\Delta < 0$, the two roots are complex-valued and distinct, $r_0 = (-b_1 - \imath\sqrt{-\Delta})/2$ and $r_1 = \bar{r}_0 = (-b_1 + \imath\sqrt{-\Delta})/2$.

1.4 Numerical Issues

1.4.1 Discriminant Nearly Zero

In practice, usually real-valued roots are desired, so we are interested in knowing when $\Delta \geq 0$. Floating-point round-off errors might lead to $\Delta < 0$ but where Δ is *nearly zero*. The vertex of the parabola corresponding to the quadratic equation is *nearly* touching the y-axis. It is your choice whether to conclude that this is close enough to a double root. The criterion will depend on your particular application. For example, you might use

```
Real discriminant = a1*a1 - 4*a0*a2;
const Real smallNonpositiveNumber = <your choice>;
if (discriminant > 0)
{
    // Return two distinct roots.
}
if (discriminant >= smallNonpositiveNumber)
{
    // Return one double root.
}
// Two complex-valued roots.  Return zero real-valued roots.
```

This code will work even when you choose the `smallNonpositiveNumber` to be zero. Instead, you might choose based on how close to zero the polynomial is,

```
Real discriminant = a1*a1 - 4*a0*a2;
const Real smallNonpositiveDiscCutoff = <your choice>;
const Real smallPositivePolyCutoff = <your choice>;
if (discriminant > 0)
{
    // Return two distinct roots.
}
if (discriminant >= smallNonpositiveDiscCutoff)
{
    // One root (double root).
    Real root = -a1/2;
    Real quadratic = a0 + root*(a1 + root*a2);
    if (fabs(quadratic) <= smallPositivePolyCutoff)
    {
        // Return one double root.
    }
}
// Two complex-value roots.  Return zero real-valued roots.
```

1.4.2 Leading Coefficient Nearly Zero

Another close-to-zero issue is when a_2 is nearly zero. Evaluation of the roots in Equation (6) becomes ill conditioned. It is quite possible that, theoretically, the roots are very large floating-point numbers or even outside the range of the floating-point representation. There is not much you can really do about this other than switch to arbitrary precision floating-point arithmetic or, considering the coefficients to be rational numbers, use root bounding and then exact rational arithmetic in a bisection scheme to compute the roots.

One possibility is to scale x to obtain a monic polynomial (rather than dividing by a_2). In the following I assume $a_2 > 0$; otherwise, multiply the equation by -1 to obtain a positive leading coefficient. Set $y = \sqrt{a_2}x$ to obtain $y^2 + (a_1/\sqrt{a_2})y + a_0 = 0$. You still have a division by a small number, but that number is larger than if you had just divided by a_2. Numerically, the square root calculation itself can be problematic–any round-off errors in computing a_2 in the first place are magnified by the square root.

Another possibility is to look at the magnitude of a_0. If this number is not close to zero, then change variables to $x = 1/y$ and compute the roots of the polynomial $g(y) = y^2 f(1/y) = a_0 y^2 + a_1 y + a_2$. For each root y of $g(y)$, the number $x = 1/y$ is a root for $f(x)$. Consider, though, that a_2 is nearly zero, so there might be a root y that is close to zero yet smaller than the minimum representable floating-point number. This is similar to the starting configuration where a root x might be larger than the maximum representable floating-point number.

1.4.3 Cancellation of Significant Digits

Also note that you can lose significant digits due to subtraction of two nearly equal floating-point numbers when a_2 is nearly zero. Specifically, consider when $a_1 > 0$. The root $(-a_1 + \sqrt{a_1^2 - 4a_0a_2})/(2a_2)$ has denominator close to zero. The square root term in the numerator is nearly a_1 when a_2 is nearly zero, so $-a_1$ plus a term that is nearly a_1 leads to cancellation of a lot of significant digits. The resulting difference is effectively noise. Worse is that the division by a very small number (a_2) is equivalent to a multiplication by a very large number ($1/a_2$) which magnifies the noise.

The remedy is to modify the root algebraically,

$$\begin{aligned}
\frac{-a_1+\sqrt{a_1^2-4a_0a_2}}{2a_2} &= \frac{-a_1+\sqrt{a_1^2-4a_0a_2}}{2a_2} \cdot \frac{-a_1-\sqrt{a_1^2-4a_0a_2}}{-a_1-\sqrt{a_1^2-4a_0a_2}} \\
&= \frac{a_1^2-(a_1^2-4a_0a_2)}{-2a_2(a_1+\sqrt{a_1^2-4a_0a_2})} \\
&= \frac{a_0}{(a_1+\sqrt{a_1^2-4a_0a_2})/2}
\end{aligned} \tag{7}$$

Now the denominator is approximately a_1, the average of a_1 and a number nearly equal to a_1, so there is no subtractive cancellation in combining a_1 and the square root term. Pseudocode for the original equations is

```
Real discriminant = a1*a1 - 4*a0*a2;
if (discriminant > 0)
{
    Real halfInvA2 = 0.5/a2, rootDiscriminant = sqrt(discriminant);
    root0 = (-a1 - rootDiscriminant)*halfInvA2;
    root1=  (-a1 + rootDiscriminant)*halfInvA2;
}
```

Pseudocode for the modified equations is

```
Real discriminant = a1*a1 - 4*a0*a2;
if (discriminant > 0)
{
    if (a1 >= 0)
    {
        float temp = -0.5*(a1 + sqrt(discriminant));
        root0 = temp/a2;
        root1 = a0/temp;
    }
    else
    {
        float temp = -0.5*(a1 - sqrt(discriminant));
        roots[0] = temp/a2;
        roots[1] = a0/temp;
    }
}
```

There is still a division by a number close to zero, but that is unavoidable. There is a trade-off here that might be an issue if your application's bottleneck is in computing roots of quadratic equations. This formulation of the algorithm requires an extra floating-point comparison and an extra division compared to the original algorithm.

Another issue with the modified algorithm is when $a_1 = 0$ and the discriminant is positive. Theoretically, the roots are $\pm r$ for some real number $r > 0$. The original algorithm will return two floating-point numbers, one the negative of the other. The modified algorithm, however, can return two distinct floating-point numbers (that are nearly identical in magnitude). For example,

```
int numRootsOriginal, numRootsModified;
float rootsOriginal[2], rootsModified[2];
float a0 = -0.01f, a1 = 0.0f, a2 = 0.001f;
QuadraticRootsOriginal(a0, a1, a2, numRootsOriginal, rootsOriginal);
QuadraticRootsModified(a0, a1, a2, numRootsModified, rootsModified);

// numRootsOriginal = 2 and numRootsModified = 2
// rootsOriginal[0] = -3.1622777f;
// rootsOriginal[1] = +3.1622777f;  // == -rootsOriginal[0]
// rootsModified[0] = -3.1622777f;
// rootsModified[1] = +3.1622775f;  // != -rootsModified[0]

float polyOriginal[2], polyModified[2];
for (int i = 0; i < 2; ++i)
{
    polyOriginal[i] = a0 + rootsOriginal[i]*(a1 + rootsOriginal[i]*a2);
    polyModified[i] = a0 + rootsModified[i]*(a1 + rootsModified[i]*a2);
}

// polyOriginal[0] = 9.4102892e-010f;
// polyOriginal[1] = 9.4102892e-010f;
// polyModified[0] = 9.4102892e-010f;
// polyModified[1] = -5.6686256e-010f;
```

Observe that the positive root for the modified algorithm has a polynomial value closer to zero than the corresponding root for the original algorithm. In this sense, the modified algorithm has a better estimate by avoiding the subtractive cancellation.

Here is another experiment.

```
float a0 = -0.01f, a1 = 0.001f, a2 = 0.000001f;

// rootsOriginal[0] = -1009.9020f;
// rootsOriginal[1] = 9.9019384f;
// rootsModified[0] = -1009.9020f;
// rootsModified[1] = 9.9019508f;

// polyOriginal[0] = -2.3621011e-008f;
// polyOriginal[1] = -1.2483159e-008f;
// polyModified[0] = -2.3621011e-008f;
// polyModified[1] = 1.6013110e-010f;
```

Once again, the modified algorithm has a better estimate of the second root. Generally, the advice is to use *root polishing* after you have estimated roots using the closed-form equations. This can be done with Newton's method or with bisection. In the previous example, one iteration of Newton's method applied to `rootsOriginal[1]` produced the root obtained from the modified algorithm.

```
int numRootsOriginal, numRootsModified;
```

```
float rootsOriginal[2], rootsModified[2];
float a0 = -0.01f, a1 = 0.001f, a2 = 0.000001f;
QuadraticRootsOriginal(a0, a1, a2, numRootsOriginal, rootsOriginal);
QuadraticRootsModified(a0, a1, a2, numRootsModified, rootsModified);

// numRootsOriginal = 2 and numRootsModified = 2
// rootsOriginal[0] = -1009.9020f;
// rootsOriginal[1] = 9.9019384f;
// rootsModified[0] = -1009.9020f;
// rootsModified[1] = 9.9019508f;

float polyOriginal[2], polyModified[2];
for (int i = 0; i < 2; ++i)
{
    polyOriginal[i] = a0 + rootsOriginal[i]*(a1 + rootsOriginal[i]*a2);
    polyModified[i] = a0 + rootsModified[i]*(a1 + rootsModified[i]*a2);
}

// polyOriginal[0] = -2.3621011e-008f;
// polyOriginal[1] = -1.2483159e-008f;
// polyModified[0] = -2.3621011e-008f;
// polyModified[1] = 1.6013110e-010f;

float poly = polyOriginal[1];
float polyderivative = a1 + 2.0f*a2*rootsOriginal[1];
rootsOriginal[1] -= poly/polyOriginal;
polyOriginal[1] = a0 + rootsOriginal[1]*(a1 + rootsOriginal[1]*a2);

// rootsOriginal[1] = 9.9019508f;
// polyOriginal[1] = 1.6013110e-010;
```

Making the leading coefficient very small shows how bad the subtractive cancellation can be.

```
float a0 = -0.01f, a1 = 0.001f, a2 = 10.0f*FLT_MIN;

// numRootsOriginal = 2 and numRootsModified = 2
// rootsOriginal[0] = -8.5070596e+033f;
// rootsOriginal[1] = 0.00000000f;  // very different from rootsModified[1]
// rootsModified[0] = -8.5070596e+033f;
// rootsModified[1] = 9.9999990f;
// polyOriginal[0] = -0.0099999998f;
// polyOriginal[1] = -0.0099999998f;
// polyModified[0] = -0.0099999998f;
// polyModified[1] = -2.5518243e-010f;

// Polish rootsOriginal[1] with one Newton iterate.
// rootsOriginal[1] = 9.9999990f;
// polyOriginal[1] = -2.5518243e-010f;
```

It is also important to observe that root polishing of the index-zero roots does not change them. This is the best you can do when the roots are so large. Great care must be taken when polishing the roots. For example, if you were to code a loop such as

```
float root = <rootsOriginal[0] or rootsModified[0]>;
float poly = a0 + root*(a1 + root*a2);
const float myPolyEpsilon = 1e-06f;
while (fabsf(poly) > myPolyEpsilon)
{
    float polyderivative = a1 + 2.0f*a2*root;
    root -= poly/polyderivative;
    poly = a0 + root*(a1 + root*a2);
}
```

you will be waiting a very long time for your program to terminate!

In this same example, you might be tempted to use bisection for root polishing. However, this will not change the results either. The floating-point numbers are *not uniformly distributed* values. The large positive floating-point numbers are sparsely distributed.

```
float a0 = -0.01f, a1 = 0.001f, a2 = 10.0f*FLT_MIN;
// root = -8.5070596e+033f (from the numerical solver)
unsigned int uRoot = *(unsigned int*)&root;  // = 0xf7d1b718
unsigned int uRootPrev = uRoot - 1;          // = 0xf7d1b717
unsigned int uRootNext = uRoot + 1;          // = 0xf7d1b719
float rootPrev = *(float*)&uRootPrev;        // = -8.5070590e+033f
float rootNext = *(float*)&uRootNext;        // = -8.5070602e+033f

float valuePrev = a0 + rootPrev*(a1 + rootPrev*a2);  // = -6.1896998e+023f
float valueRoot = a0 + root*(a1 + root*a2);          // = -0.0099999998f;
float valueNext = a0 + rootNext*(a1 + rootNext*a2);  // = +6.1897012e+023f
```

The last three lines of the pseudocode show the polynomial values for three consecutive 32-bit floating-point numbers. The value at the root is on the order of 10^{-2}. The values at the adjacent floating-point numbers is on the order of 10^{23}. I believe the root finder did the best it can do!

To see how structuring the polynomial expression itself can be influential,

```
valuePrev = a0 + rootPrev*a1 + (rootPrev*rootPrev)*a2;  // = -6.1896998e+023f
valueRoot = a0 + root*a1 + (root*root)*a2;              // = 0.00000000f;
valueNext = a0 + rootNext*a1 + (rootNext*rootNext)*a2;  // = +6.1897012e+023f
```

The evaluation of `valueRoot` has a problem with subtractive cancellation. The expressions `a0 + root*a1` and `(root*root)*a2` evaluate to the same magnitude floating-point number, one positive and one negative, so they cancel to zero. At first glance, the problem appears to be that the intermediate product `root*root` exceeds the maximum representable 32-bit floating-number. However, internally the compiler generates code for high-precision registers, so this is not the problem. If you switch to using double-precision explicitly to evaluate `valueRoot`, you still get a result of identically zero. Perhaps `a0+root*(a1+root*a2)` is producing an inaccurate result? In fact, it is accurate. Using exact rational arithmetic and converting the result back to a floating-point number produces the value `-0.0099999998f`.

1.4.4 Root Deflation

Given a real-valued root r_0 to the quadratic, you can factor the quadratic into

$$a_2x^2 + a_1x + a_0 = (x - r_0)(a_2x - b) = a_2x^2 - (b + r_0a_2)x + br_0 \tag{8}$$

This process is called *root deflation.* Compute a root to a polynomial and factor out a linear term to obtain a polynomial of degree one less than what you started with. You can continue the process to find all the roots. Generally, each step introduces round-off errors, so you must consider polishing the roots at the end (or after each step and at the e nd).

In the quadratic example, he linear coefficient implies $b = -(a_1 + r_0a_2)$ and the other root is $r_1 = b/a_2 = -(a_1 + r_0a_2)/a_2 = -a_1/a_2 - r_0$. Numerical round-off errors can affect the construction in this manner, so you shoud probably polish the root estimate r_1. In the example with `a0 = -0.01`, `a1 = 0.001`, and `a2 = 10*FLT_MIN`, the deflation leads to $r_1 = 0$, regardless of whether you use $-(a_1 + r_0a_2)/a_2$ or $-a_1/a_2 - r_0$. In another example,

```
float a0 = -0.01f, a1 = 0.001f, a2 = 0.000001f;

// From the original algorithm.
// float root0  = -1009.9020f
// float value0 = -2.3621011e-008f
// float root1  = +9.9019384f
// float value1 = -1.2483159e-008f
float root1a = -(c1 + root*c2)/c2;  // = +9.9019270f
float root1b = -(c1/c2 + root);     // = +9.9019279f
float value1a = c0 + root1a*(c1 + root1a*c2);  // = -2.4153890e-008f
float value1b = c0 + root1b*(c1 + root1b*c2);  // = -2.3181329e-008f;
```

The modified algorithm still produces a better estimate of the root. But as always, you can polish the root using Newton iterates.

2 Cubic Roots

The general cubic equation is of the form

$$f(x) = a_3x^3 + a_2x^2 + a_1x + a_0 = 0 \tag{9}$$

where $a_3 \neq 0$. The cubic polynomial is said to be *monic* when the third-degree coefficient is one,

$$f(x) = x^3 + b_2x^2 + b_1x + b_0 = 0 \tag{10}$$

The monic cubic equation is obtained from the general equation by dividing by a_3, in which case $b_i = a_i/a_3$ for $i = 0, 1, 2$.

The analysis that leads to the equations for the roots of Equation (10) are somewhat lengthy. First, a change of variables is made to eliminate the squared term. Let $x = y - b_2/3$ and $g(y) = f(x)$; then

$$g(y) = y^3 + c_1 y + c_0 = 0 \tag{11}$$

where $c_1 = (3b_1 - b_2^2)/3$ and $c_0 = (2b_2^3 - 9b_1 b_2 + 27b_0)/27$.

2.1 Constant Term is Zero

If $c_0 = 0$, the Equation (11) becomes $y^3 + c_1 y = 0$ and the roots are 0 and $\pm\sqrt{-c_1}$. The corresponding x-roots are obtained by subtracting $b_2/3$ from the y-roots.

2.1.1 One Triple Real-Valued Root

If $c_1 = 0$, then $r_0 = 0$ is a triple root.

2.1.2 Three Simple Real-Valued Roots

If $c_1 < 0$, then there are three distinct real-valued roots: $r_0 = 0$, $r_1 = -\sqrt{-c_1}$, and $r_2 = +\sqrt{-c_1}$.

2.1.3 One Simple Real-Valued Root, Two Simple Complex-Valued Roots

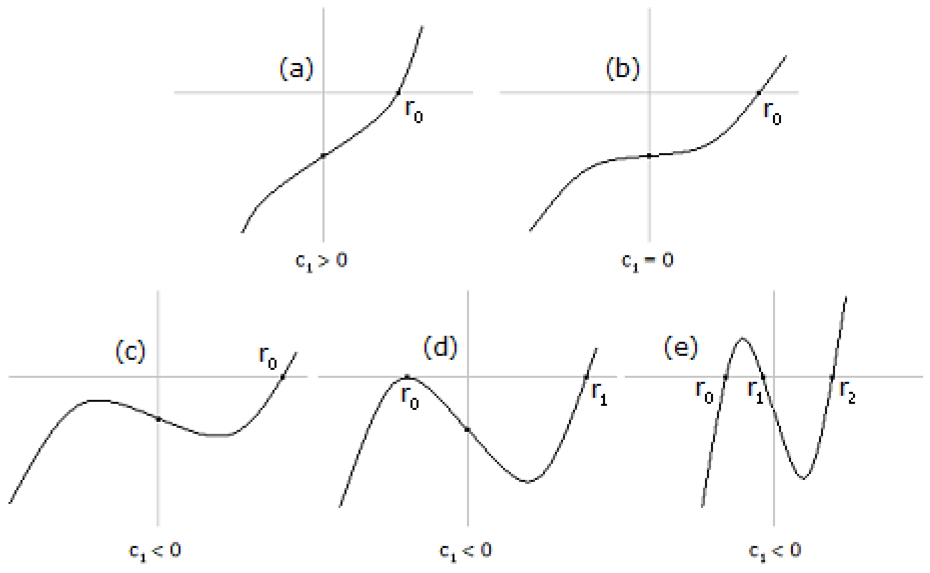
If $c_1 > 0$, then there is one real-valued root and two complex-valued roots: r_0, $r_1 = -\imath\sqrt{c_1}$, and $r_2 = +\imath\sqrt{c_1}$.

2.2 Constant Term is Not Zero

Let $c_0 \neq 0$. We can focus our attention on the case $c_0 < 0$. For if $c_0 > 0$, the change of variables $z = -y$ and $h(z) = -g(-y)$ leads to $h(z) = z^3 + c_1 z + (-c_0)$ where $-c_0 < 0$. The polynomial $h(z)$ has negative constant term, which is the case we analyze here.

Let us plot the graph of $g(y) = y^3 + c_1 y + c_0$, where $c_0 < 0$. The first derivative of the polynomial is $g'(y) = 3y^2 + c_1$ and the second derivative of the polynomial is $g''(y) = 6y$. The graph has a point of inflection when $g''(y) = 0$, so this point occurs at $y = 0$. Figure 2.1 shows five graph configurations that depend on the sign and value of c_1,

Figure 2.1 Several graph configurations for $g(y)$ when $c_0 < 0$.



2.2.1 One Simple Real-Valued Root, One Double Real-Valued Root

In Figure 2.1(d), the graph of $g(y)$ is tangent to the y-axis, so the polynomial has a double root at the point of contact. This occurs at r_0 for which $g(r_0) = 0$ and $g'(r_0) = 0$. The second equation may be solved for r_0^2, substituted in the first equation, after which several algebraic operations leads to

$$r_0 = \frac{-3c_0}{2c_1} \tag{12}$$

The polynomial must factor to $(y - r_0)^2(y - r_1)$, where r_1 is a simple real-valued root:

$$y^3 + c_1 y + c_0 = (y - r_0)^2(y - r_1 = y^3 - (2r_0 + r_1)y^2 + (r_0^2 + 2r_0 r_1)y - r_0^2 r_1 \tag{13}$$

The second-degree term on the right-hand side must be zero to match the left-hand side, so

$$r_1 = -2r_0 = \frac{6c_0}{2c_1} \tag{14}$$

This special case of a double root shows that c_0 and c_1 are related algebraically. From $g'(r_0) = 0$, we have $r_0 = (-c_1/2)^{1/2}$. Substituting this into $g(r_0) = 0$ and performing a few algebraic operations leads to

$$\Delta = \left(\frac{c_0}{2}\right)^2 + \left(\frac{c_1}{3}\right)^3 = 0 \tag{15}$$

where the first equality defines Δ, called the *discriminant* of the cubic polynomial. We will see this quantity later in the document.

2.2.2 One Simple Real-Valued Root, Two Simple Complex-Valued Roots

In Figure 2.1(c), the derivative of the polynomial is zero in two locations, $\pm d$, where $d = \sqrt{-c_1/3}$ is obtained by solving $0 = g'(d) = 3d^2 + c_1$. At the location $-d$, the polynomial is negative:

$$0 > g(-d) = -d^3 - c_1 d + c_0 = \tfrac{-2c_1 d}{3} + c_0 = 2\left(\tfrac{-c_1}{3}\right)^{3/2} + c_0 \tag{16}$$

Therefore,

$$0 < \left(\frac{-c_1}{3}\right)^{3/2} < \frac{-c_0}{2} \tag{17}$$

The positivity of the left-hand side guarantees that squaring does not change the direction of the inequality, so

$$\left(\frac{-c_1}{3}\right)^{3} < \left(\frac{-c_0}{2}\right)^{2} \tag{18}$$

This equation shows that $\Delta > 0$ in the configuration of Figure 2.1(c). In fact, $c_1 \geq 0$ in the configurations of Figure 2.1(a) and Figure 2.1(b). In both cases, $\Delta > 0$. Therefore, the condition $\Delta > 0$ guarantees that the cubic polynomial has one simple real-valued root and two simple complex-valued roots.

The real-valued root can be constructed using the following observation. A real number may be written as the sum of two real numbers, so the root r_0 may be written as $r_0 = u + v$, where u and v are determined by the condition that r_0 is a root. Specifically, $r_0^2 = u^2 + 2uv + v^2$ and

$$r_0^3 = u^3 + 3u^2 v + 3uv^2 + v^3 = 3uv(u+v) + u^3 + v^3 = (3uv)r_0 + (u^3 + v^3) \tag{19}$$

We also know

$$0 = r_0^3 + c_1 r_0 + c_0 = (3uv + c_1) r_0 + (u^3 + v^3 + c_0) \tag{20}$$

It is sufficient to compute u and v for which $uv = -c_1/3$ and $u^3 + v^3 = -c_0$. Cubing the first expression, we have $u^3 v^3 = (-c_1/3)^3$. Define $\alpha = u^3$ and $\beta = v^3$; then, $\alpha\beta = (-c_1/3)^3$ and $\alpha + \beta = -c_0$. Solving the second equation for β and substituting in the first,

$$\alpha^2 + c_0 \alpha + (-c_1/3)^2 = 0 \tag{21}$$

We could have easily chosen to solve for β, which leads to the same quadratic equation. This is a monic quadratic equation with roots

$$\alpha = -c_0/2 - \sqrt{\Delta}, \quad \beta = -c_0/2 + \sqrt{\Delta} \tag{22}$$

where Δ is the discriminant defined by Equation (15). We know that $\Delta > 0$ for the cases under consideration, so α and β are real-valued and the (principal) cube roots are real-valued. We conclude that

$$r_0 = u + v = \alpha^{1/3} + \beta^{1/3} = \left(-c_0/2 - \sqrt{\Delta}\right)^{1/3} + \left(-c_0/2 + \sqrt{\Delta}\right)^{1/3} \tag{23}$$

The complex-valued roots are obtained by factoring the polynomial knowing the simple real-valued root,

$$y^3 + c_1 y + c_0 = (y - r_0)(y^2 + r_0 y + (r_0^2 + c_1)) = 0 \tag{24}$$

The quadratic term has roots

$$r = \frac{-r_0 \pm \sqrt{-(c_1 + 3r_0^2)}}{2} \tag{25}$$

Notice that $c_1 + 3r_0^2 = g'(r_0) > 0$, so $-(c_1 + 3r_0^2) < 0$ and the other roots are indeed (non-real) complex-valued.

2.2.3 Three Simple Real-Valued Roots

The polynomial can have three distinct real-valued roots, as illustrated in Figure 2.1(e), and is characterized by $\Delta < 0$. The construction in the previous section for $r_0 = u + v$ still applies, but the values α and β of Equation (22) are now complex-valued,

$$\alpha = -c_0/2 - \boldsymbol{\imath}\sqrt{-\Delta}, \;\; \beta = -c_0/2 + \boldsymbol{\imath}\sqrt{-\Delta} \tag{26}$$

The polar representation in the complex plane of the second root is

$$\beta = -c_0/2 + \boldsymbol{\imath}\sqrt{-\Delta} = \rho(\cos\theta + \boldsymbol{\imath}\sin\theta) = \rho\exp(\boldsymbol{\imath}\theta) \tag{27}$$

where ρ is the length of $\beta = \xi + \boldsymbol{\imath}\eta$ as a 2-tuple in the $\xi\eta$-plane and θ is the angle formed with the positive ξ-axis ($\theta > 0$ corresponds to a counterclockwise rotation from this axis). There are three cube roots,

$$\beta^{1/3} = \rho^{1/3}\exp(\boldsymbol{\imath}\theta/3), \;\; \rho^{1/3}\exp(\boldsymbol{\imath}(\theta/3 + 2\pi/3)), \;\; \rho^{1/3}\exp(\boldsymbol{\imath}(\theta/3 - 2\pi/3)) \tag{28}$$

The cube roots of α are the complex conjugates of the cube roots of β,

$$\alpha^{1/3} = \rho^{1/3}\exp(-\boldsymbol{\imath}\theta/3), \;\; \rho^{1/3}\exp(-\boldsymbol{\imath}(\theta/3 + 2\pi/3)), \;\; \rho^{1/3}\exp(-\boldsymbol{\imath}(\theta/3 - 2\pi/3)) \tag{29}$$

The real-valued polynomial roots are the sums of the cube roots and their complex conjugates,

$$\begin{aligned} r_0' &= \rho^{1/3}\exp(\boldsymbol{\imath}\theta/3) + \rho^{1/3}\exp(-\boldsymbol{\imath}\theta/3) \\ &= 2\rho^{1/3}\cos(\theta/3) \end{aligned} \tag{30}$$

and

$$\begin{aligned} r_1' &= \rho^{1/3}\exp(\boldsymbol{\imath}(\theta/3 + 2\pi/3)) + \rho^{1/3}\exp(-\boldsymbol{\imath}(\theta/3 + 2\pi/3)) \\ &= 2\rho^{1/3}\cos(\theta/3 + 2\pi/3) \\ &= \rho^{1/3}(\cos(\theta/3) - \sqrt{3}\sin(\theta/3)) \end{aligned} \tag{31}$$

and

$$\begin{aligned} r_2' &= \rho^{1/3}\exp(\boldsymbol{\imath}(\theta/3 - 2\pi/3)) + \rho^{1/3}\exp(-\boldsymbol{\imath}(\theta/3 - 2\pi/3)) \\ &= 2\rho^{1/3}\cos(\theta/3 - 2\pi/3) \\ &= \rho^{1/3}(\cos(\theta/3) + \sqrt{3}\sin(\theta/3)) \end{aligned} \tag{32}$$

Once computed, these roots may be sorted to produce the ordered roots $\{r_0, r_1, r_2\}$ as shown in Figure 2.1.

2.3 Summary of Real-Valued Root Construction

For the monic polynomial equation, $y^3 + c_1 y + c_0 = 0$, define $\Delta = (c_0/2)^2 + (c_1/3)^3$.

If $\Delta > 0$, there is exactly one real-valued root

$$r_0 = \left(-c_0/2 - \sqrt{\Delta}\right)^{1/3} + \left(-c_0/2 + \sqrt{\Delta}\right)^{1/3} \tag{33}$$

If $\Delta = 0$ and $c_0 = 0$, then $c_1 = 0$ and $r_0 = 0$ is a triple root. If $\Delta = 0$ and $c_0 \neq 0$, then there are two real-valued roots, one of them a double root.

$$r_0 = \frac{-3c_0}{2c_1} \text{ (double root)}, \quad r_1 = -2r_0 \tag{34}$$

If $\Delta < 0$, there are three distinct real-valued roots. Compute $\mu = (-c_1/3)^{3/2}$ and $\phi = \operatorname{atan2}(\sqrt{-\Delta}, -c_0/2)/3$. The real-valued roots are

$$r_0 = 2\mu\cos(\phi), \quad r_1 = \mu(\cos(\phi) + \sqrt{3}\sin(\phi)), \quad r_2 = \mu(\cos(\phi) - \sqrt{3}\sin(\phi)) \tag{35}$$

2.4 Numerical Issues

3 Quartic Roots

The general quartic equation is of the form

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0 \tag{36}$$

where $a_4 \neq 0$. The quartic polynomial is said to be *monic* when the fourth-degree coefficient is one,

$$f(x) = x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = 0 \tag{37}$$

The monic cubic equation is obtained from the general equation by dividing by a_3, in which case $b_i = a_i/a_3$ for $i = 0, 1, 2$.

The Taylor polynomial of $f(x)$ centered at λ is

$$f(x) = f(\lambda) + f'(\lambda)(x-\lambda) + \frac{1}{2}f''(\lambda)(x-\lambda)^2 + \frac{1}{6}f'''(\lambda)(x-\lambda)^3 + \frac{1}{24}f''''(\lambda)(x-\lambda)^4 \tag{38}$$

where the derivatives of f are

$$\begin{aligned} f'(x) &= 4x^3 + 3b_3x^2 + 2b_2x + b_1 \\ f''(x) &= 12x^2 + 6b_3x + 2b_2 \\ f'''(x) &= 24x + 6b_3 \\ f''''(x) &= 24 \end{aligned} \tag{39}$$

Choose $\lambda = -b_3/4$ so that $f'''(\lambda) = 0$. Define $y = x - \lambda$ and $g(y) = f(x)$,

$$g(y) = y^4 + c_2y^2 + c_1y + c_0 \tag{40}$$

where

$$\begin{aligned} c_0 &= f(-b_3/4) = \tfrac{-3b_3^4 + 16b_2b_3^2 - 64b_1b_3 + 256b_0}{256} \\ c_1 &= f'(-b_3/4) = \tfrac{b_3^3 - b_2b_3 + 2b_1}{2} \\ c_2 &= f''(-b_3/4) = \tfrac{-3b_3^2 + 8b_2}{4} \end{aligned} \tag{41}$$

The first three derivatives of $g(y)$ are

$$\begin{aligned} g'(y) &= 4y^3 + 2c_2 y + c_1 \\ g''(y) &= 12y^2 + 2c_2 \\ g'''(y) &= 24y \end{aligned} \tag{42}$$

3.1 Special Cases

3.1.1 Constant Coefficient is Zero

If $c_0 = 0$, a root is $r = 0$ and the equation reduces to a cubic polynomial, which we solved previously. For the remainder of this section, it is assumed that $c_0 \neq 0$, in which case any real-valued root cannot be zero.

3.1.2 Linear Coefficient is Zero

If $c_0 \neq 0$ and $c_1 = 0$, then the polynomial is $g(y) = y^4 + c_2 y^2 + c_0$, which is quadratic in y^2. The squared roots are formally

$$r^2 = \frac{-c_2 \pm \sqrt{c_2^2 - 4c_0}}{2} \tag{43}$$

If $\Delta = c_2^2 - 4c_0 < 0$, there are no real-valued roots. The right-hand side of Equation (43) is complex valued, and taking the square root produces only complex-valued numbers. Equivalently, $\Delta < 0$ implies $g(y) > 0$ for all y.

If $\Delta = 0$, then it is necessary that $c_2 \leq 0$ for there to be real-valued roots. When $c_2 = 0$, it must be that $c_0 = 0$ for $\Delta = 0$, but we already know that the root is $r = 0$ and has multiplicity 4. When $c_2 < 0$, the roots are $r = \pm\sqrt{-c_2/2}$. Each root has multiplicity 2. If $\Delta > 0$, the right-hand side of Equation (43) produces two real-valued numbers. For each nonnegative number, the square roots of that number are roots to g.

3.1.3 Quadratic Coefficient is Zero

If $c_0 \neq 0$, $c_1 \neq 0$, and $c_2 = 0$, then the polynomial is $g(y) = y^4 + c_1 y + c_0$ and has derivatives $g'(y) = 4y^3 + c_1$ and $g''(y) = 12y^2 \geq 0$. The second derivative is nonnegative, which implies that g is a convex function. It has either two distinct real-valued roots, one real-valued root of multiplicity 2, or no real-valued roots.

To have a repeated real root, it is necessary that $g(r) = 0$ and $g'(r) = 0$. We know that $r \neq 0$, so $g''(r) \neq 0$ which implies the multiplicity is 2. Observe that $0 = 4g(r) - rg'(r) = 3c_1 r + 4c_0$, so the root must be

$$r = \frac{-4c_0}{3c_1}$$

Evaluation of $g'(r) = 0$ leads to the condition

$$\Delta = \left(\frac{c_1}{4}\right)^4 - \left(\frac{c_0}{3}\right)^3 = 0$$

This condition may be used to distinguish among the cases. If $\Delta < 0$, the equation has no real-valued roots. If $\Delta = 0$, the equation has a real-valued root of multiplicity 2 at $r = -4c_0/(3c_1)$. If $\Delta > 0$, the equation has two distinct real-valued roots. A construction for the distinct roots is not immediately obvious.

Introduce a parameter z that is to be determined later. Consider

$$(r^2 + z)^2 = r^4 + 2zr^2 + z^2 = -c_1 r - c_0 + 2zr^2 + z^2 = 2zr^2 - c_1 r + (z^2 - c_0)$$

The right-hand side is a quadratic polynomial in r. Its discriminant is $\delta = c_1^2 - 8z(z^2 - c_0)$. Choose z so that $\delta = 0$. This gives us a cubic polynomial equation, $z^3 - c_0 z - c_1^2/8 = 0$, which has a positive real-valued root. This root allows us to factor the quadratic polynomial,

$$(r^2 + z)^2 = 2z\left(r - \frac{c_1}{4z}\right)^2$$

Taking square roots,

$$r^2 + z = \pm\sqrt{2z}\left(r - \frac{c_1}{4z}\right)$$

which leads to two quadratic equations,

$$r^2 + \sqrt{2z}\,r + \left(z - \frac{c_1}{2\sqrt{2z}}\right) = 0 \tag{44}$$

and

$$r^2 - \sqrt{2z}\,r + \left(z + \frac{c_1}{2\sqrt{2z}}\right) = 0 \tag{45}$$

The discriminant for Equation (44) is $\Delta_0 = -2z + 2c_1/\sqrt{2z}$. When $c_1 < 0$, $\Delta_0 < 0$ and the quartic equation does not have real-valued roots due to this term. The discriminant for Equation (45) is $\Delta_1 = -2z - 2c_1/\sqrt{2z}$. When $c_1 > 0$, $\Delta_1 < 0$ and the quartic equation does not have real-valued roots due to this term. It is not possible for both discriminants to be nonnegative, so the quartic cannot have four real-valued roots. At most one discrimant is nonnegative (two real-valued roots) or both are negative (no real-valued roots).

3.2 General Cases

Previously, we analyzed several cases where one or more polynomial coefficients are zero. In the general case of this section, we assume that none of the coefficients are zero; that is, $c_0 \neq 0$, $c_1 \neq 0$, and $c_2 \neq 0$.

3.2.1 Nonsimple Roots

A root r of multiplicity at least 2 occurs when $g(r) = 0$ and $g'(r) = 0$. Based on our previous assumption that $c_0 \neq 0$, a root r cannot be zero. Define $h(y) = 4g(y) - yg'(y)$, which also has the root r,

$$0 = h(r) = 4g(r) - rg'(r) = 2c_2 r^2 + 3c_1 r + 4c_0 \tag{46}$$

Define $\ell(y) = c_2 g'(y) - 2yh(y)$, which also has the root r,

$$0 = \ell(r) = c_2 g'(r) - 2rh(r) = -6c_1 r^2 + (2c_2^2 - 8c_0)r + c_1 c_2 \tag{47}$$

Define $m(y) = 3c_1h(y) + c_2\ell(y)$, which also has the root r,

$$0 = m(r) = 3c_1h(r) + c_2\ell(r) = \left[\frac{1}{3}\left(27c_1^2 + 8c_2^3\right) - \frac{2c_2}{3}\left(c_2^2 + 12c_0\right)\right] r + c_1\left(c_2^2 + 12c_0\right) = d_1r + d_0 \tag{48}$$

The factorization of $m(r)$ was made to emphasize the terms $(27c_1^2 + 8c_2^3)$ and $(c_2^2 + 12c_0)$, which occur later in the discussion. The coefficients d_0 and d_1 are defined by the last equality. If $d_1 \neq 0$, then the root is $r = -d_0/d_1$.

3.2.2 Real-Valued Root of Multiplicity 3

Such a root r occurs when $g(r) = 0$, $g'(r) = 0$, $g''(r) = 0$, but $g'''(r) \neq 0$. The last equation implies $r \neq 0$. This condition and the equation $g''(r) = 0$ imply $c_2 < 0$ is a necessary condition for a root of multiplicity 3.

Equations (46), (47), and (48) already show us how to compute the repeated root. However, we have an addition constraint, namely, $g''(r) = 0$. Define $n(y) = 3g'(y) - yg''(y)$, which also has the root r,

$$0 = n(r) = 3g'(r) - rg''(r) = 4c_2r + 3c_1 \tag{49}$$

Define $p(y) = 2h(y) - y\ell(y)$, which also has the root r,

$$0 = p(r) = 2h(r) - r\ell(r) = 3c_1r + 8c_0 \tag{50}$$

The previous two equations show that the root is

$$r = \frac{-3c_1}{4c_2} = \frac{-8c_0}{3c_1} \tag{51}$$

A necessary condition for r to be a triple root is obtained by solving $g''(r) = 0$ for $r^2 = -c_2/6$ and substituting $r = -3c_1/(4c_2)$ to obtain

$$0 = 27c_1^2 + 8c_2^3 \tag{52}$$

We may also substitute $r^2 = -c_2/6$ into $g(r) = 0$ to obtain

$$0 = c_2^2 + 12c_0 \tag{53}$$

These terms occur in Equation (48). For the case of a triple root, $d_0 = d_1 = 0$, so it is not possible to solve Equation (48) for r; the construction in this section was necessary instead. It is easy enough to verify that Equations (52) and (53) are also sufficient conditions for a triple root.

Given a triple root r, the other root ρ has multiplicity 1. It may be constructed by observing that

$$y^3 + c_2y^2 + c_1y + c_0 = (y - r)^3(y - \rho) = y^4 - (\rho + 3r)y^3 + 34(\rho + r)y^2 - r^2(r + 3\rho)y + r^3\rho \tag{54}$$

The third-degree terms must match, so $\rho = -3r$.

3.2.3 Real-Valued Root of Multiplicity 2

Because the root is only a double root, it is not possible for both $(27c_1^2 + 8c_2^3)$ and $(c_2^2 + 12c_0)$ to be zero (they are both zero only for triple roots). We know that the root cannot be zero, so in Equation (48), it must be that $c_1 \neq 0$ and $c_2^2 + 12c_0 \neq 0$; that is, $d_0 \neq 0$. In turn, this implies that $d_1 \neq 0$. The root must be $r = -d_0/d_1$.

In the construction, there is only one choice for r. However, it is possible that $g(y)$ has two distinct real-valued roots, both of multiplicity 2. This appears to be contradictory, but in fact it is not. The polynomial factors into $(y-r)^2(y^2+v_1y+v_0)$. When you expand this and match coefficients with $y^4+c_2y^2+c_1y+c_0$, you obtain $v_1 = 2r$, $v_0 - 3r^2 = c_2$, $2r^3 - 2rv_0 = c_1$, and $r^2v_0 = c_0$. Then $y^2+v_1y+v_0 = y^2+2ry+v_0$. To be a squared quantity, we would need $v_0 = r^2$, in which case the final factorization is $(y-r)^2(y+r)^2$. However, then $c_2 = v_0 - 3r^2 = -2r^2$, $c_1 = 2r^3 - 2rv_0 = 0$, and $c_0 = r^4$. In this section we have assumed $c_1 \neq 0$, because the case of $c_1 = 0$ was handled previously. Thus, with our constraints on the polynomial coefficients, the factor $y^2+v_1y+v_0$ has no real-valued roots.

3.2.4 The General Construction

The root construction is similar to the special case when $c_2 = 0$. Introduce a parameter z that will be determined later. Consider

$$(r^2+z)^2 = r^4 + 2zr + z^2 = -c_2r^2 - c_1r - c_0 + 2zr^2 + z^2 = (2z-c_2)r^2 - c_1r + (z^2-c_0)$$

The right-hand side is a quadratic polynomial with discriminant $\delta = c_1^2 - 4(2z-c_2)(z^2-c_0)$. This is a cubic polynomial that has at least one real-valued root. Choose z to be such a root; then

$$(r^2+z)^2 = (\alpha r - \beta)^2$$

for α and β dependent on z and the coefficients of the quartic polynomial. This leads to two quadratic equations

$$r^2 + z \pm (\alpha r - \beta) = 0$$

If either equation has a nonnegative discriminant, the quartic will have corresponding real-valued roots.

4 Real Parts of Polynomial Roots

Let $P_n(z) = \sum_{k=0}^{n} a_k^{(n)} z^k$ be a polynomial of degree n with complex coefficients. Let the roots be $z_j^{(n)}$, $j = 1, \ldots, n$. Define the $n-1$ degree polynomial

$$P_{n-1}(z) = [a_n^{(n)}\bar{a}_{n-1}^{(n)} + a_{n-1}^{(n)}\bar{a}_n^{(n)} - a_n^{(n)}\bar{a}_n^{(n)}z]P_n(z) + \left(a_n^{(n)}\right)^2 z \sum_{k=0}^{n}(-1)^{n-k}\bar{a}_k^{(n)}z^k = \sum_{k=0}^{n-1} a_k^{(n-1)} z^k$$

where $\bar{c}$ denotes the complex conjugate of c. Let the roots be $z_j^{(n-1)}$, $j = 1, \ldots, n-1$. Let $\mathrm{Re}(c)$ denote the real part of c. The Routh-Hurwitz criterion states:

$$\mathrm{Re}(z_j^{(n)}) < 0,\ 1 \le j \le n \text{ if and only if } \mathrm{Re}(a_{n-1}^{(n)}/a_n^{(n)}) > 0 \text{ and } \mathrm{Re}(z_j^{(n-1)}) < 0,\ 1 \le j \le n-1.$$

This gives a recursive way of deciding if all the real parts of a polynomial are negative.

I use this criterion for applications where $P_n(z)$ is the characteristic polynomial for a real symmetric $n \times n$ matrix M. Necessarily the roots are all real, so the criterion allows us to decide if the polynomial has all negative or all positive real roots. Thus, M is negative definite if all roots of $P_n(z)$ are negative, and M is positive definite if all roots of $P_n(z)$ are positive.

For polynomials with all real coefficients we have the recurrence relations

$$\begin{aligned} a_0^{(n-1)} &= 2a_0^{(n)} a_{n-1}^{(n)} a_n^{(n)} \\ a_k^{(n-1)} &= a_n^{(n)} [2a_{n-1}^{(n)} a_k^{(n)} - a_n^{(n)} a_{k-1}^{(n)} (1 + (-1)^{n-k})], \quad 1 \le k \le n-1 \end{aligned}$$

The code is a direct implementation of these relations, except that the coefficients of the polynomials are adjusted so that the leading coefficient is 1. Note that all roots of $P(z)$ have positive real parts if and only if all roots of $P(-z)$ have negative real parts.

For quadratic polynomials $P(z) = z^2 + az + b$, all roots have negative real part if and only if $a > 0$ and $b > 0$. For cubic polynomials $P(z) = z^3 + az^2 + bz + c$, all roots have negative real part if and only if $a > 0$, $ab - c > 0$, and $c > 0$. For quartic polynomials $P(z) = z^4 + az^3 + bz^2 + cz + d$, all roots have negative real part if and only if $a > 0$, $ab - c > 0$, $d > 0$, and $c(ab - c) > a^2 d$.