Polynomial Zeros

The fundamental theorem of algebra tells us that a polynomial p of degree n has exactly n zeros (if you count right), but it doesn't tell us how to compute them. We have formulas (using just arithmetic and n^{th} roots) for the zeros of polynomials of degree 1, 2, 3 and 4. The quest for formulas for the zeros of higher degree polynomials culminated in a remarkably beautiful theory (now called Galois theory) which among other things tells us that

there can be no such formula for the roots of polynomials of degree more than 4.

Even if there were such formulas, they probably wouldn't be a good way to *compute* the zeros, but such a theorem means polynomial root finding must, in general, be an iterative process.

Since this a ubiquitous problem and since polynomials have such structure, special methods taking advantage of the structure are usually employed. While we certainly could throw Newton's, secant or bisection at this problem, we want to take advantage of the special structure of polynomials. Generalizing Newton's method by approximating p by a quadratic (or higher) Taylor polynomial is an obvious step. Let x^* be a root of p, and x_n an approximation to x^*. If we write $p^* = p_n + h$, then

$$p(x^*) = p(x_n) + hp'(x_n) + \frac{h^2}{2}p''(x_n) + \mathrm{O}(h^3).$$

Dropping the $\mathrm{O}(h^3)$ term and noting that $p(x^*) = 0$ yields a quadratic equation in h

$$t(h) = \left(\frac{p''(x_n)}{2}\right) h^2 + p'(x_n)h + f(x_n) = 0,$$

and letting be h_1 the root of t nearest 0 gives the update $x_{n+1} = x_n + h_1$. This idea requires 3 function evaluations per iteration and has order of convergence $\alpha = 3$ per iteration, and $\alpha = \sqrt[3]{3} \approx 1.44$ per function evaluation.

As we used secant to rescue Newton's, we use *Müller's* to rescue this Taylor method. There is exactly one polynomial of degree 2 or less passing through any 3 points in the plane (with distinct x-coordinates). Let $M(x)$ be the one passing through the points $(x_{n-2}, p(x_{n-2}))$, $(x_{n-1}, p(x_{n-1}))$, and $(x_n, p(x_n))$. If we now let m_1 be the root of M nearest x_n we have the Müller update $x_{n+1} = m_1$.

Müller's method requires only one function evaluation per iteration ($p(x_{n-1})$ and $p(x_{n-2})$ were computed earlier), and has order of convergence $\alpha \approx 1.84$.

Once one root, say r_1, of p is computed, we can divide p by $(x - r_1)$ to get a new polynomial $p_2(x) = p(x)/(x - r_1)$ of degree $n-1$. This process is called *deflation*. Each time a new root, say r_j of p_j is computed, we can compute the polynomial $p_{j+1}(x) = p_j(x)/(x - r_j)$. After $n - 1$ deflation steps, we have all of the roots of p.

We leave you with a question: *What about complex roots?*
And a confession: deflation, as described here, *doesn't work in finite precision.*