Polynomial Roots

John Kormylo

Newton-Rhapson finds zero crossings for a function using the same approach as Newton's method for finding a local optima. From a first order Taylor's series approximaton

$$f(x) \approx f(x(k)) + f'(x(k))(x - x(k))$$

we obtain the iterative algorithm to set

$$f(x(k+1)) = 0$$

using

$$x(k+1) = x(k) - [f'(x(k))]^{-1} f(x(k)) \quad .$$

Note that this works when $x$ and $f(x)$ are vectors and $f'(x)$ is a (hopefully invertable) matrix.

One could use this algorithm to solve polynomial roots one at a time then remove them from the polynomial. I find it more effective to solve all of the roots simultaneously using the vector function given by

$$f_i(x) = p(x_i) / \prod_{j \neq i} (x_i - x_j) \quad \text{for} \quad i = 1, 2, \ldots, n$$

where $p(s)$ is the $n$th order polynomial to be factorized. The advantage is that as the estimates become more accurate, $f(x)$ becomes closer to being linear and the algorithm will converge faster (exponential convergence).

For an initial guess I use complex points evenly spaced around the unit circle of the form

$$x_k = \exp \sqrt{-1}(\frac{2\pi k}{n} + \phi) \quad \text{for} \quad k = 1, 2, \ldots, n$$

where $\phi$ is a small offset to prevent the roots from forming complex pairs (typically $\phi = \pi/2n$). After a single iteration the various $x_k$ components have obviously locked onto their respective roots.

Interestingly, this algorithm can handle double roots (or worse) with no modification. During convergence, the estimates $x_i$ are "attracted" to the roots but "repelled" by each other. Consequently, two components of $x$ will approach a double root from opposite directions and will converge more slowly than single roots. The stopping criterion will always be met before double roots converge to the point of causing a division by zero (unless the polynomial is quadratic).