Newton’s method. See also my web site on Newton Basins at http://aleph0.clarku.edu/~djoyce/newton/newton.html.

Newton’s method for functions of one variable. Newton’s method is a method to approximate solutions to equations of the form \( f(x) = 0 \), that is, how to find roots of functions \( f : \mathbb{R} \rightarrow \mathbb{R} \). Not only is the method easy to comprehend, it is a very efficient way to find the solution to the equation.

It’s an iterative process that starts with a guess \( x_0 \), finds a better approximation \( x_1 \) based on the value of \( x_0 \), then finds an ever better approximation \( x_2 \) based on the value of \( x_1 \), and continues until the desired accuracy is achieved.

The process is illustrated in figure 1. Let \( x_0 \) be the initial function. Look at the tangent to the graph of the function at the point \( (x_0, f(x_0)) \) and follow that down to the \( x \)-axis to find the next approximation \( x_1 \). Algebraically, that means you set \( x_1 \) to the value

\[
x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.
\]

Frequently \( x_1 \) will be closer to a root than \( x_0 \) was. More estimates can be found recursively by

\[
x_{k+1} = x_i - \frac{f(x_k)}{f'(x_k)}.
\]

The sequence \( x_0, x_1, x_2, \ldots, x_k, \ldots \) often approaches a root of the function \( f \).

Newton’s method for complex functions. The Newton Basins web site uses a generalization from real variables to complex variables. For that, a function is a complex function \( f : \mathbb{C} \rightarrow \mathbb{C} \). Derivatives are defined in the same way for a complex function as they are for a real function, so the recursive formula

\[
x_{k+1} = x_i - \frac{f(x_k)}{f'(x_k)}
\]

makes sense and works for complex functions, too. (For background on complex numbers, see Dave’s Short Course on Complex Numbers.)

In particular, the web site examines roots of polynomial functions of complex variables. All the complex numbers that lead to one particular root form what is called a Newton basin for the root.

For example, consider the polynomial with roots \( i, -i \), and 1.75,

\[
f(z) = (z-i)(z+i)(z-1.75) = z^3 - 1.75z^2 + z - 1.75.
\]

Figure 3 shows the three Newton basins for this cubic polynomial. The image is the a square in the complex plane where the real part varies from \(-2\) to \(2\) and the imaginary part varies from \(-2i\) to \(2i\). The root \( i \) is in the center of the light red circle, the root \(-i\) is in the center of the light green circle, and the root 1.75 is in the center of the light blue circle.
If your first approximation lies fairly near the root $z = i$, then the succeeding approximations approach $i$ very fast. Likewise if the first approximations are near either of the other two roots, the approximations approach that root quickly, too. Note, however, if you start about midway between two of the roots, then you might hop into the main basin of the third root. The boundaries between the three basins are fractals.

Figure 3 shows the Newton basins in small portion of the complex plane for the cubic polynomial with roots $z = 1$, $z = -1.384609 - 0.9i$, and $z = 0.384609 + 0.93i$.

**Newton’s method for vector fields.** In this course we can look at a generalization to vector fields, $f : \mathbb{R}^n \to \mathbb{R}^n$. With our new concept of derivative, we can write the recursion relation as

$$x_{k+1} = x_k - (Df(x_k))^{-1}f(x_k)$$

where $x \in \mathbb{R}^n$. The total derivative $Df(x_k)$ of $f$ at $x_k$ is an $n \times n$ matrix, and $(Df(x_k))^{-1}$ is the inverse matrix, should the inverse exist.

The text has some examples illustrating how to use this form to solve simultaneous nonlinear equations.

This generalization to multivariate functions includes the previous generalization to complex numbers since a complex function can be interpreted as a function $f : \mathbb{R}^2 \to \mathbb{R}^2$.

Math 131 Home Page at

http://math.clarku.edu/~djoyce/ma131/