Finding the roots of a Function

Often in Physics one has to solve an algebraic equation that does not have a simple form. One can usually cast the equation into the form

$$f(x) = 0 \tag{1}$$

The goal is to find the value(s) of x that satisfy this equation, i.e. the roots of $f(x)$. If there is no analytic solution, then numerical methods can be used to find a solution to the desired accuracy. In this chapter, we will describe three different numerical algorithms for finding the roots of $f(x)$: the Bisection Method, the "Bracket and Half" method, and Newton's method. In each method, an algorithm is developed for producing a series of $x's$, x_i. The algorithm is designed such that the series x_i converges to a root of $f(x)$.

Bisection Method

The bisection method finds roots in the interval between the values a and b, where $a < b$. If $f(a)$ and $f(b)$ have opposite signs, then there is at least one root of $f(x)$ between a and b. Another way of stating that $f(a)$ and $f(b)$ have opposite signs is $f(a)f(b) < 0$. The next point in the series is the mid-point of a and b, $c = (a + b)/2$. If $f(c)$ and $f(a)$ have opposite signs, then a root is between a and c. If $f(c)$ and $f(b)$ have opposite signs, then a root is between c and b. Only one of these options will be true, since $f(a)$ and $f(b)$ have themselves opposite signs. The interval that contains the root is bisected again and the procedure repeated.

Below is an example code that carries out the bisection method:

```
for $(i = 1; i < imax; i++)$
{
c=(a+b)/2;
if $(f(c) * f(a) < 0)$ then b = c;
if $(f(c) * f(b) < 0)$ then a = c;
}
```

One could use an "else" statement instead of the two "if" statements. The approximate root is c, with an estimated error of $(b - a)/2$.

There are some problems to watch out for. Eventually $f(a)$, $f(b)$, and $f(c)$ will be very close to zero. When multiplying two of these quantites, the product can fall below the accuracy of the machine. Instead of a "for" loop, a "while" loop can be used:

```
c = (a+b)/2;
tol=0.0000001;
while ($f(c) > tol$)
{
c=(a+b)/2;
if ($f(c) * f(a) < 0$) then b = c;
if ($f(c) * f(b) < 0$) then a = c;
}
```

where "tol" can be determined from the accuracy desired or the machine tolerance. Instead of $f(c)$ being the condition, the while loop could continue till $|b - a|$ is less than a tolerance.

One limitation of the bisection method is that one needs to choose a and b such that there is a root between them. The next method avoids this requirement, and only needs a starting value and step size.

Bracket and Half Method

The bracket and half method is best explained by describing the algorithm. One choses an initial starting value, x_1, and an initial step size δ. The next value of x, $x_2 = x_1 + \delta$. If $f(x_2)f(x_1) > 0$, then we have not passed a root. The value of δ doesn't change. We continue until $f(x_{i+1})f(x_i) < 0$. When this occurs, we have steped over the root. Now we halve the step size and reverse its direction. The process is continued until we have reached the desired accuracy.

Below is an example code for the bracket and half method:

```
a=starting value
del=initial step size
for ($i = 1; i < imax; i++$)
{
b = a + del;
if ($f(b) * f(a) < 0$) then del=-del/2;
a = b;
}
```

The bracket and half method is useful for finding the zeros above (or below) a certain value of x. It is also useful in finding the bound state energy levels of the non-relativistic Schroedinger equation For this application, the algorithm is refered as the "shooting method".

Newton's Method

Newton's method can be used if the first derivative, $f'(x)$, is known. We will just state the algorithm here. For a derivation as to its validity, refer to a text on numerical methods.

One starts with an initial guess for the root of $f(x)$, x_1. The algorthym is an equation to obtain x_{i+1} from x_i:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{2}$$

Under certain conditions, the series x_i will converge to a root of $f(x)$. Usually in physics applications one does not know the analytic form of $f'(x)$, so Newton's method is not commonly used.