Chapter 4

Roots of Algebraic Equations

In this chapter we will

- learn how to solve algebraic equations using the
  - Bisection method
  - Newton-Raphson method
  - Secant method
- introduce the concept of convergence of iterative methods

One of the most obvious uses of computer algorithms in the physical and mathematical sciences is in solving algebraic equations. A computational method often provides the only way of obtaining the roots of an equation, since analytic solutions exist only for a handful of cases. In other situations, a computational method may provide a faster route in obtaining the roots of an equation that has known analytic solutions.

In this chapter, we will consider general equations of the form

$$f(x) = 0\ , \tag{4.1}$$

where $f(x)$ is any function of the unknown variable x. Every equations can be trivially put in this form by subtracting its right-hand side from its left-hand side. The function $f(x)$ may be an analytic expression in closed form, such as

$$f(x) = xe^x - 1\ . \tag{4.2}$$

Alternatively, it may be a complicated computational procedure that returns a real number for each value of the argument x. In this chapter, we will consider only equations that involve real quantities and search for roots that are real numbers.

Root Multiplicity

A root x_0 of an equation $f(x) = 0$ is said to have multiplicity k if there is a function $g(x)$ such that

$$f(x) = (x - x_0)^k g(x)\ . \tag{4.3}$$

Alternatively, a root x_0 of an equation $f(x) = 0$ is said to have a multiplicity k if the $k-$th order derivative of the function $f(x)$ evaluated at x_0 is zero, i.e., if

$$\left.\frac{d^k f(x)}{dx^k}\right|_{x_0} = 0\ . \tag{4.4}$$

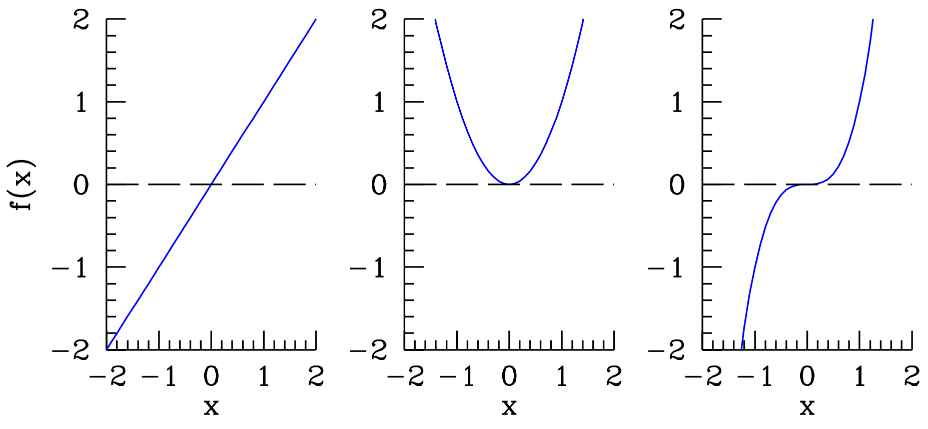


Figure 4.1: An example of functions that have a root of multiplicity *(left)* one, *(center)* two, and *(right)* three.

Figure 4.1 shows a few illustrative examples of functions with roots of multiplicity one, two, and three. In general, a double root corresponds to a function that is tangent to the $x-$axis, whereas a root of multiplicity three or higher corresponds to a function that has an inflection point on the $x-$axis.

4.1 The Bisection Method

This is the simplest and most robust method for finding the root of an equation, provided that we can guess an initial interval in which one and only one root exists. The algorithm successively divides the interval in half (bisects) keeping the solution within its limits, until it reaches a desired level of accuracy. The main drawback of the method is its slow rate of convergence.

The bisection algorithm is based on the intermediate value theorem:

Intermediate Value Theorem

Consider a function $f(x)$ that is continuous in the interval $[a,b]$ with $f(a)\cdot f(b)<0$. Then, there exists a value x_0 in the interval $[a,b]$ such that $f(x_0)=0$.

In other words, this theorem states that, if a continuous function $f(x)$ changes sign in the interval $[a,b]$, then there is at least one root of the equation $f(x)=0$ within this interval.

It is worth paying close attention to the requirement that the function $f(x)$ is continuous in the interval $[a,b]$. If it is not, then the theorem does not guarantee the existence of a solution. Consider, for example, the function

$$f(x)=\frac{1}{x-2}\;. \tag{4.5}$$

Clearly, the function changes sign in the interval $[0,4]$. However, we cannot apply the intermediate value theorem for this function, because it is discontinuous at $x=2$. Indeed, there exists no value of x for which $f(x)=0$.

When using the bisection algorithm to find iteratively the solution of an equation of the form $f(x)=0$ we take the following steps:

1. We start with an initial interval $[a,b]$ in which we know *a priori* that the solution exists

Roots of Polynomials

A polynomial of degree n

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 = 0$$

with real coefficients has at most n real roots. If the degree is an odd number, then the polynomial is guaranteed to have at least one real root.
A polynomial of degree 2 is called a quadratic; a polynomial of degree 3 is called a cubic; a polynomial of degree 4 is called a quartic. General solutions for polynomials of degree up to four exist in closed form. It can be proved, however, that there exist no general solutions for polynomials of degree 5 or higher. This theorem was proved in the early 19th century by the Norwegian mathematician Niels Hendrik Abel (1802–1829) and the Italian mathematician Paolo Ruffini (1765–1822).

2. We calculate the midpoint of the interval

$$x_{\rm mid} = \frac{1}{2}(a+b) . \tag{4.6}$$

3. At this point, we need to decide whether the solution lies in the interval $[a, x_{\rm mid}]$ or in the interval $[x_{\rm mid}, b]$. We will use the intermediate value theorem to make this decision. If

$$f(a) \cdot f(x_{\rm mid}) < 0 , \tag{4.7}$$

then the function changes sign in the interval $[a, x_{\rm mid}]$ and, therefore, this is where the solution lies. In this case, we set $b = x_{\rm mid}$ so that, at the end of this iteration, the solution lies again in the interval $[a, b]$. If, on the other hand,

$$f(a) \cdot f(x_{\rm mid}) > 0 , \tag{4.8}$$

then we set $a = x_{\rm mid}$ for the same reason.

4. If the width of the interval $(b - a)$ is still larger than the required accuracy, we go back to step #2, to refine the interval.

5. When the width of the interval $(b - a)$ becomes smaller than the required accuracy, then we have reached the solution to the equation.

In step #3, we need to pay special attention to what the algorithm does if the midpoint of the interval happens to be very close to the actual solution, i.e., when $f(x_{\rm mid}) = 0$ to within numerical accuracy. In principle, since the algorithm has reached the solution to the equation, it can simply exit and return this value. However, this will require the introduction of an additional condition in the main body of the algorithm, which will slow down its execution considerably. Alternatively, we can change the condition (4.7) in step #3 to

$$f(a) \cdot f(x_{\rm mid}) \le 0 \tag{4.9}$$

and proceed with the rest of the algorithm unaltered.

It is important also to emphasize here the fact that the bisection method does not guarantee that the equation will be satisfied to any accuracy. Indeed, the algorithm only guarantees that the root of an equation is bracketed to the required

```
double foo(double x);              // prototype for function f(x)

void bisection(double *lower, double *upper)
/* Uses the bisection algorithm to find the solution to an algebraic
   equation of the form f(x)=0 that is known to lie in the interval
   [lower,upper]. The form of the equation is provided by the
   function foo(x) and is supplied by the user. The parameter
   ACCURACY determines the level to which the initial interval will
   be refined by successive bisections.
*/
#define ACCURACY 1.0e-6
{
  double xmid;                     // the midpoint of the interval
  double a=*lower,b=*upper;        // its bounds

  do
    {
      xmid=0.5*(a+b);              // calculate midpoint
      if (foo(a)*foo(xmid)<=0.0)   // if solution is in lower half
        b=xmid;                    // set upper bound=midpoint
      else                         // otherwise
        a=xmid;                    // set lower bound=midpoint
    }
  while (fabs(b-a)>ACCURACY);      // until converged to ACCURACY

  *lower=a;                        // return the bounds
  *upper=b;

  return;
}
```

accuracy. We can calculate what this implies for the value of the function $f(x)$ by evaluating its Taylor expansion from $x = a$ to $x = b$,

$$f(b) = f(a) + \left.\frac{df}{dx}\right|_{x=a} (b-a) + ... \tag{4.10}$$

Neglecting the terms of higher order and rearranging the above expression, we obtain

$$|f(b) - f(a)| = \left|\frac{df}{dx}\right|_{x=a} |b-a| \; . \tag{4.11}$$

This last equality shows that, at the end of the bisection algorithm, the equation will be satisfied to the required accuracy only if the function $f(x)$ has a weak dependence on the variable x, i.e., if $|df/dx| \le 1$. If we need to circumvent this problem, we can revise step #3 by requiring not only that the interval $(b-a)$ is smaller than the required accuracy, but that the value of the function evaluated at the midpoint, $f(x_{\text{mid}})$ is smaller than the accuracy, as well.

A C function that implements the bisection method as discussed here is shown above. This function requires two arguments, the lower and upper bounds of the initial guess for the interval in which a unique solution is guaranteed to exist. Upon completion, the arguments of the function contain the lower and upper bounds of an interval with a width that is determined by the parameter `ACCURACY` that contains the root of the equation. This algorithm assumes that the function $f(x)$ is provided by the user as a C function of the form

```
double foo(double x);
```

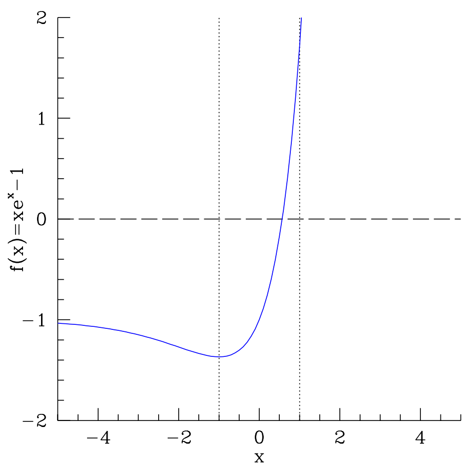


Figure 4.2: The function $f(x) = xe^x - 1$ plotted against the variable x. A quick glance reveals that the equation $f(x) = 0$ has a unique solution in the interval $[-1, 1]$.

Guessing the Initial Interval

Finding the initial interval in which a unique solution to an equation lies is more of a guess than an algorithm. In most cases, calculating the limiting values of the function at $\pm\infty$ or examining the graph of the function gives the best handle on bracketing its solutions. These two techniques are illustrated with the following example.

Example: Solving the equation $xe^x - 1 = 0$ with the bisection method

In order to study whether this equation has roots, we will first check the limits of the function $f(x) = xe^x - 1$ as $x \to \pm\infty$. Calculating one of the limits is simple, because

$$\lim_{x\to\infty} (xe^x - 1) = +\infty \ . \tag{4.12}$$

Calculating the limiting value of the function as $x \to -\infty$, however, requires the use of L'Hospital's rule

$$\lim_{x\to-\infty} (xe^x - 1) = \lim_{x\to-\infty} (xe^x) - 1 = \lim_{x\to-\infty} (\frac{x}{e^{-x}}) - 1 = -1 \ . \tag{4.13}$$

Because the two limits have opposite signs and the function is continuous, the equation has at least one solution for some real value of the variable x.

If the function is monotonic and has two limits of opposing signs, then it has at most one solution. We study the monotonicity of the function by evaluating its derivative

$$\frac{df}{dx} = (x+1)e^x \ . \tag{4.14}$$

This is positive when $x > -1$ and negative when $x < -1$. At $x = -1$, the function takes the value $f(-1) \simeq -1.368$, which is negative. As a result, between $x = -\infty$ and $x = -1$ the function is negative and decreasing, it has a minimum at $x = -1$, and between $x = -1$ and $x = +\infty$ the function is increasing and changes sign. Therefore, the equation has only one solution in the interval $[-1, +\infty)$.

In order to obtain a finite upper bound on the interval in which the solution exists, we recognize the fact that if $x > 1$,

$$f(x) = xe^x - 1 > 1 \cdot e^1 - 1 > 1 \ . \tag{4.15}$$

The function, therefore, changes sign in the interval $[-1, 1]$ in which the single root of the equation lies.

Alternatively, we can guess an initial interval for the bisection algorithm by plotting the function and studying graphically its behavior. A quick glance at the figure shows that the equation has only one solution, which lies in the interval $[-1, 1]$, as we inferred earlier using analytical methods.

Inserting this interval as an initial guess in a bisection function allows us to find the solution to the equation as $x_0 = 0.567143$ with an accuracy of 1×10^{-6}.

Assessment of the Bisection Method

The bisection method is a very robust and simple to implement algorithm for finding the root of an equation. It guarantees finding the solution at the required accuracy, within a finite number of steps. The method, however, has a number of disadvantages:

(i) It cannot find the solutions to all equations. For example, a polynomial that has a root of multiplicity 2 does not change sign across the solution. More specifically, the equation

$$(x-2)^2 = 0 \tag{4.16}$$

cannot be solved with the bisection method because the function $f(x) = (x-2)^2$ is positive or zero for all real values of x. The same is also true for many equations that are not polynomials of x. For example, the equation

$$e^x - x - 1 = 0 \tag{4.17}$$

has an obvious root at $x = 0$, even though the function $f(x) = e^x - x - 1$ is never negative.

(ii) It requires an a priori knowledge of an interval in which a unique solution exists. This not necessarily a disadvantage of the method, because it forces us to study thoroughly the equation before we solve it and, hence, protects us from making mistakes. It is worth asking, however, which root does the bisection method find when there are more than one roots in the initial interval. The answer, of course, depends on the particular equation and the choice of initial interval. One can answer this question probabilistically, however, considering a large number of application of the method to various equations with various initial intervals[1]. The result is fascinating!

Consider an equation $f(x) = 0$ that has N simple roots in the interval $[a, b]$, which we will denote by $x_1 < x_2 < ... < x_N$. The bisection algorithm will find the odd numbered roots (x_1, x_3, etc.) with equal probability and the even numbered roots (x_2, x_4, etc.) with zero probability. It is fair to say that these are not good chances to play with!

(iii) It converges slowly. Although the uncertainty in the solution is reduced by a factor of two after each iteration, the bisection method is the slowest among the root finding algorithms that we discuss in this chapter. Before embarking into a more detailed study of this issue, however, we need to discuss in general the order of convergence of an iterative method.

4.2 Order of Convergence of an Iterative Method

When we employ an iterative method to solve a problem, we start with an initial guess of the solution and improve it systematically with each cycle of the algorithm.

Our aims is to reach a given accuracy for the solution with the smallest number of cycles. We often measure the efficiency of an iterative method to achieve this by calculating its order of convergence.

For concreteness, we will consider an iterative method that aims to find the root of an algebraic equation x_0 and denote by x_i, the approximate solution after i iterations. We will also use the symbol ϵ_i to denote the fractional error between the approximate and the correct solution after i iterations. If a constant number λ exists such that

$$\lim_{i\to\infty} \frac{|x_{i+1}-x_0|}{|x_i-x_0|^N} = \lambda \tag{4.18}$$

then we say that our iterative method converged with an order N and an asymptotic error constant λ. If $N = 1$, we call the convergence linear. If $N = 2$, we call the convergence quadratic.

The higher the order of convergence of an iterative method, the faster the solution converges to a required accuracy. Indeed, after the first few iterations, the definition (4.18) implies that the error in the solution achieved by a method with order of convergence N is

$$\epsilon_{i+1} \simeq \lambda \epsilon_i^N \ . \tag{4.19}$$

We can use this definition to calculate the order of convergence of the bisection method. We will denote by $\Delta \equiv b - a$ the size of the initial interval in which the single root exists. This is also an estimate of the uncertainty ϵ_0 with which we know the solution *a priori*. After one iteration, the interval and hence the uncertainty is reduced to half, i.e., $\epsilon_1 = \Delta/2$. After two iterations, it is reduced by one more factor of two, i.e., $\epsilon_2 = (\Delta/2)/2 = \Delta/2^2$, and so on. In general,

$$\epsilon_{i+1} = \frac{1}{2}\epsilon_i \tag{4.20}$$

and, therefore, the bisection method converges linearly with an asymptotic error constant of 0.5.

We can use the linear order of converge of the bisection method to calculate how many iterations we will need to complete in order to reach an uncertainty δ in the final estimate of the solution. It is easy to show by induction that after i iterations the uncertainty is $\epsilon_i = \Delta/2^i$. Setting this to δ and solving for i we obtain

$$i = \frac{\log_{10}\Delta - \log_{10}\delta}{\log_{10} 2} \ . \tag{4.21}$$

We can, therefore, reduce the uncertainty with which we know the solution to an algebraic equation by six orders magnitude by using the bisection method for 20 iterations.

4.3 The Newton-Raphson Method

The Newton-Raphson method is an iterative algorithm for solving algebraic equations of the form $f(x) = 0$. It overcomes two disadvantages of the bisection method. It does not require knowledge of an interval in which a unique solution lies and it converges quadratically. However, it does not guarantee convergence to a solution and requires an analytic knowledge of the first derivative of the function $f(x)$.

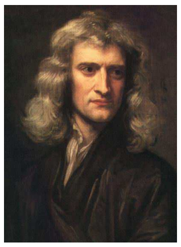
In order to introduce this method, we will consider a function $f(x)$ that is continuous and such that the equation $f(x) = 0$ has a root at $x = x_0$. We will also assume that we have made an initial guess for the value of the root, which we will

The Development of the Newton-Raphson Method[2]

Sir Isaac Newton, the famous English scientist of the 17th century, used in his notes a method introduced by the French mathematician François Viète (1540–1603) in order to solve polynomial equations of order higher than four. He then followed a similar approach in his famous book *Philosophiae Naturalis Principia Mathematica* to solve the non-linear equation

$$x - e\sin(x) = M$$

that connects the eccentric anomaly x to the mean anomaly M of a planet in an orbit of eccentricity e.
In 1690, the English mathematician Joseph Raphson (approximately 1648–1715) published a similar method to solve polynomial equations of order up to ten, with some references to Newton's work. Neither Newton nor Raphson used the newly invented techniques of calculus to derive the method. Instead, they used algebraic and geometric arguments. Half a century later, Thomas Simpson (1710–1761) presented a similar method for solving non-linear equations based on arguments of calculus. The familiar Newton-Raphson method was published in 1798 by the French mathematician Joseph Louis Lagrange (1736–1813) with reference to the work by Newton and Raphson but not to the work by Simpson.


Isaac Newton
(1643–1727)

denote by x_i, which is different from the true solution by an amount $\delta = x_0 - x_i$. Using this information, the Newton-Raphson method allows us to improve our initial guess by calculating approximately the value of δ.

In order to estimate δ, we Taylor expand the function $f(x)$ from x_i to the true solution $x_0 = x_i + \delta$. We obtain

$$f(x_0) = f(x_i + \delta) = f(x_i) + f'(x_i)\delta + \frac{1}{2}f''(x_i)\delta^2 + \ldots\,, \tag{4.22}$$

where primes denote derivatives of the function with respect to the variable x. By definition, $f(x_0) = 0$ and hence we can solve the above equation for δ. Neglecting terms of order δ^2 or higher we obtain

$$\delta = -\frac{f(x_i)}{f'(x_i)}\,. \tag{4.23}$$

This is the amount we need to add to our guess in order to approach the true solution. Because we have neglected terms of second and higher order in the Taylor expansion, however, we most probably did not reach the solution. We, therefore, need to repeat these steps until we have converged to an acceptable solution.

We will consider the solution converged if two separate criteria are satisfied. First, the correction introduced to the solution by the last iteration should be smaller than a required accuracy. Second, the value of the function at the current value of the solution should be zero, to within a required accuracy.

Figure 4.3 illustrates graphically the Newton-Raphson algorithm. In this example, x_1 is our first guess for the root of the equation $f(x) = 0$. When we approximate the function around x_1 with the Taylor expansion (4.22) while keeping only terms up to first order, we are effectively replacing the function $f(x)$ with the straight line A_1B_1 that is tangent at $f(x_1)$. We then find the abscissa x_2 of the point at which the straight line A_1B_1 crosses the $x-$axis by evaluating the correction δ in

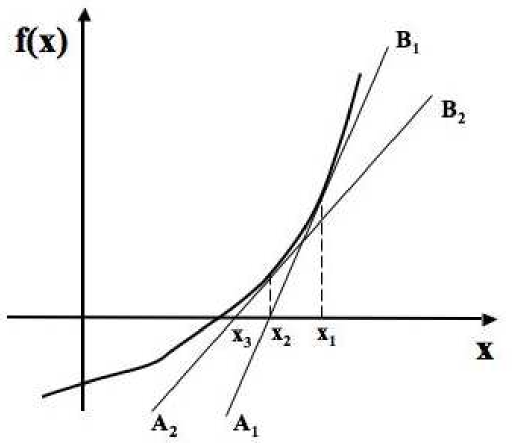


Figure 4.3: A graphical representation of the Newton-Raphson algorithm for finding the root of an algebraic equation.

equation (4.23). We repeat this procedure by approximating the function around x_2 with the tangent line A_2B_2, find the point x_3 at which this tangent line crosses the $x-$axis, and continue until we have reached the true solution with the required accuracy. It is often said that following the Newton-Raphson method is equivalent to sliding down the function along tangent lines until we find the solution.

We can summarize the steps needed to find the root of an equation with the Newton-Raphson method as follows:

1. We start with an initial guess to the root, x_i
2. We update our guess setting $x_i + \delta \rightarrow x_i$ where the correction δ is given by expression (4.23).
3. If δ is smaller than the required accuracy and $f(x_i) = 0$ to within the required accuracy then we we have reached the solution. Otherwise, we go back to step #2.

There is a particular issue that we need to be careful about when calculating the correction δ using equation (4.23). If the current estimate of the solution, x_i, is at or near an extremum or an inflection point of the function $f(x)$, then the value of the derivative $f'(x)$ becomes very small and, therefore, the correction δ becomes very large. Graphically, the tangent to the function becomes nearly horizontal and sends the next estimate of the root towards $+\infty$ or $-\infty$. There is no good, general way of recovering from such an unfortunate step. The best course of action is to use a different initial guess for the solution.

A C function that implements the Newton-Raphson method as discussed here is shown above. This function requires one arguments, the initial guess for the solution. Upon completion, it returns the root to the equation. This algorithm assumes that the function $f(x)$ and its derivative $f'(x)$ are provided by the user as C functions of the form

```
double foo(double x);
```

and

```
double fooprime(double x);
```

If the solution to an equation has multiplicity one, then the Newton-Raphson

Order of Convergence

```
double foo(double x);                   // prototypes for functions
double fooprime(double x);

double newtonraphson(double root)
/* Uses the Newton-Raphson method to find the solution to an
   algebraic equation of the form f(x)=0. The function f(x) and its
   derivative are provided by the functions foo(x) and fooprime(x)
   and are supplied by the user. The parameter ACCURACY is used in
   determining whether the method has converged to a solution. Upon
   completion, it returns the root of the equation.
 */
#define ACCURACY 1.0e-6
{
  double delta;                         // Newton-Raphson correction
  double fx,fprime;                     // f(x) and its derivative

  fx=foo(root);                         // f(x) at x=root
  do
    {
      fprime=fooprime(root);            // f'(x) at x=root
      if (fabs(fprime)>ACCURACY)        // if not extremum/inflection
        delta=-fx/fprime;               // calculate correction
      else                              // otherwise
        {                               // print error message and exit
           printf("Newton-Raphson cannot converge\n");
           return 0.0;
        }
      root+=delta;                      // update root
      fx=foo(root);                     // f(x) at root
    }                                   // until converged
  while (fabs(delta)>ACCURACY || fabs(fx)>ACCURACY);

  return root;
}
```

method converges quadratically. In order to show this, we will consider two successive approximations x_i and x_{i+1} evaluated after the i-th iteration and the one following it. These two trials are related by equation (4.23) or

$$x_{i+1} - x_i = -\frac{f(x_i)}{f'(x_i)} \ . \tag{4.24}$$

If we denote by $\epsilon_i \equiv x_i - x_0$ and $\epsilon_{i+1} \equiv x_{i+1} - x_0$ the deviations of the two approximations from the true solution x_0, then

$$\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)} \ . \tag{4.25}$$

We now Taylor expand the function $f(x)$ around the true solution x_0 to obtain

$$\begin{aligned} f(x) &= f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + ... \\ &= f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + ... \ , \end{aligned} \tag{4.26}$$

where we have used the fact that $f(x_0) = 0$. We then take the derivative of the expanded function with respect to the variable x and evaluate it at x_i to obtain

$$f'(x_i) = f'(x_0) + f''(x_0)(x_i - x_0) + ... \tag{4.27}$$

Inserting equations (4.26) and (4.27) into equation (4.25) we obtain

$$\epsilon_{i+1} = \epsilon_i - \frac{f'(x_0)\epsilon_i + \frac{1}{2}f''(x_0)\epsilon_i^2}{f'(x_0)[1 + \frac{f''(x_0)\epsilon_i}{f'(x_0)}]} \ . \tag{4.28}$$

Note that this last equation is value only if $f'(x_0) \neq 0$, which is the reason why we required the multiplicity of the solution to be one.

We now expand the denominator in the fraction using the approximate relation

$$(1+x)^a \simeq 1 + ax + ... \tag{4.29}$$

that is valid when $x \ll 1$, rearrangement some terms keeping only those that are up to second order in the small parameter ϵ_i and arrive at the final result

$$\epsilon_{i+1} = \epsilon_i^2 \left[\frac{f''(x_0)}{2f'(x_0)}\right] \ . \tag{4.30}$$

This last relation proves that the Newton-Raphson method converges quadratically to a solution of multiplicity one of an algebraic equation. This is, of course, true as long as the initial guess is not significantly different than the true solution.

If the multiplicity of the root is equal to $m > 1$, then it can be shown that the Newton-Raphson method converges only linearly with an asymptotic error constant of

$$l = \frac{1}{1-m} \ . \tag{4.31}$$

Assessment of the Newton-Raphson method

In general, the Newton-Raphson method is an algorithm that converges fast to the solution of an equation starting from a simple initial guess. It also does not require an *a priori* knowledge of an interval in which the solution exists. This method, however, also has a number of disadvantages.

(i) It requires an analytic knowledge of the derivative $f'(x)$. In many situations this may not be possible, if the function $f(x)$ itself is the result of a numerical calculation. In other cases, calculating the derivative may be computationally so expensive that it counters the gain in speed achieved by the quadratic convergence of the method.

(ii) It does not converge globally. Indeed, our proof of quadratic convergence relies on two assumptions. That the current guess to the solution is not very difference from the true value and that the derivative of the function $f'(x)$ at the guess is not zero. If either of these conditions is not satisfied, the method may not converge at all to the true solution.

(iii) In very symmetric situations, it enters a cyclic behavior without converging to a solution.

Further Reading

1. *Which root does the bisection algorithm find?* Corliss, G., SIAM Review **19**, 2, 325 (1977)

2. *Historical Development of the Newton-Raphson Method*, Ypma, T. J., SIAM Review **37**, 4, 531 (1995)