Chapter 5

Finding Roots and Maxima

5.1 Introduction

One of the most basic tasks of mathematics applied to scientific problem solving is that of finding the zeroes of a function, or equivalently, the roots of an equation. Many situations require finding the roots of a single equation or a system of equations.

For example, the location of the extremal points of a function requires finding the zeroes of the derivatives of that function. Many problems which involve critical paths also require the solution of algebraic equations, such as determining all the ray paths that are possible in a complex optical system.

The solution of nonlinear systems of algebraic equations is frequently not possible using formal analytic methods, as it is for linear systems. Hence it is necessary to resort to iterative numerical methods to solve such systems. It is at first surprising how many scientific problems result in the need to solve nonlinear equations, and how difficult it is to solve these equations, or even to determine whether solutions exist at all. The great variety of forms for such equations means that scientists usually need familiarity with a number of different methods to handle such problems. In this chapter we will concentrate on methods for the solution of a single nonlinear equation in one variable.

5.2 Bisection method

The first class of methods we discuss can be called *bracketing methods* because one starts with the knowledge that a zero lies in some interval, so that it is only necessary to refine the knowledge of the interval so that the interval containing the zero has a length that is smaller than the required precision for the zero. One of the simplest methods is the first *bisection* or *interval-halving method* of solving the equation $f(x) = 0$. As long as we know that $f(x)$ is continuous on some interval $[x_a, x_b]$, and $f(x_a)$ and $f(x_b)$ have different signs,

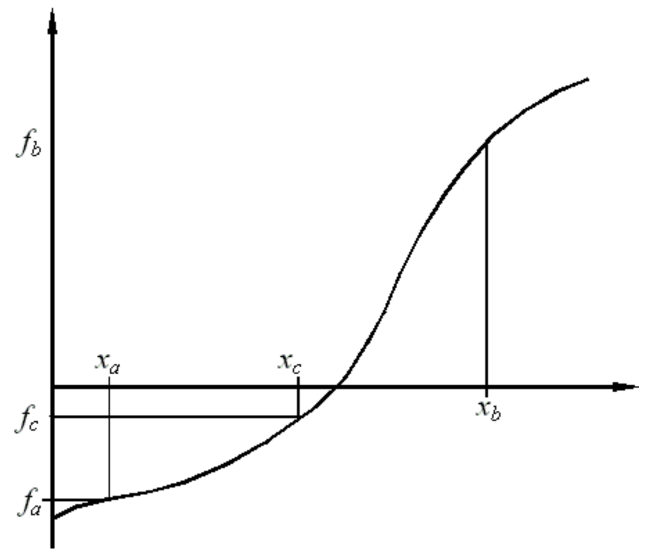

Figure 5.1: Successive steps in the bisection method to find the zero of $f(x)$.

there is at least one value of x in the interval $[x_a, x_b]$ for which $f(x) = 0$. We then perform the bisection step by evaluating $f(x_c)$ at $x_c = (x_a + x_b)/2$. Either $f(x_c) = 0$, in which case we have found a root, or $f(x_c)$ has the same sign as either $f(x_a)$ or $f(x_b)$. We then discard the half interval for which there is no sign change, and repeat bisection for the remaining interval. In this manner we progressively refine the interval that contains the sign change until we have bracketed the required root to the required level of precision. See Fig. 5.1 for an illustration and § 5.7.1 for the algorithm.

5.3 Regula falsi method or solution by linear interpolation

The bisection method is very simple, but generally quite inefficient, in part because it only makes use of the sign of the function $f(x)$ at each evaluation, while ignoring its magnitude. Thus it ignores significant information which could be used accelerate the finding of the root. A method based on interpolation makes use of this information by approximating the function on the interval $[x_a, x_b]$ by the chord joining the points $(x_a, f(x_a))$ and $(x_b, f(x_b))$, that is, the straight line given by

$$\frac{y - f(x_a)}{x - x_a} = \frac{f(x_b) - f(x_a)}{x_b - x_a} \,. \tag{5.1}$$

This method is also known as the *regula falsi method*. It is also a bracketing method, since it also refines the definition of the interval containing the zero. Solving the linear equation for $y = 0$ yields the new interval endpoint within the interval $[x_a, x_b]$:

$$x_c = x_a - f(x_a)\frac{x_b - x_a}{f(x_b) - f(x_a)} \tag{5.2}$$

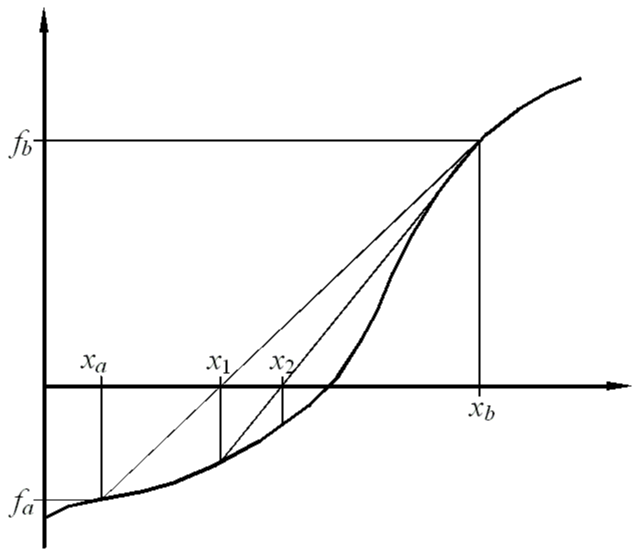

Figure 5.2: Successive steps in the regula falsi iteration to find the zero of $f(x)$. Notice that in this example the right hand end of the test interval remains fixed.

The choice between the two intervals $[x_a, x_c]$ and $[x_c, x_b]$ is made by evaluating $f(x_c)$ and discarding the interval whose endpoints have the same sign, as was done in the bisection method. This iteration process is repeated, but it will converge more quickly than the bisection method, since the information about the magnitude of $f(x)$ pushes the x_c value more quickly towards the actual root. The iteration is continued until the end points of the interval show no change to the required precision in subsequent iterations. One important point to note is that frequently the interval does not become small, as it did in the bisection method, since one end may remain fixed while the other end closes in on the zero from one side only. The example shown in Figure 5.2 illustrates this effect. See § 5.7.1 for the algorithm.

Figure 5.2 illustrates the regula falsi method and makes clear one fault with the method. If the function $f(x)$ is strongly curved on the starting interval, the approach to the root can be one-sided in the sense that one end of the interval remains fixed rather than the endpoint that is discarded oscillating from one end to the other. This can be partially suppressed by artificially decreasing the value of $f(x)$ at the stagnant end of the interval to $f(x)/2$ at each iteration. This does not alter the sign of $f(x)$ at that endpoint, but does tend to accelerate the convergence towards the root by inducing a more equal distribution of the interval end which is dropped.

The real danger with this method is when $f(a)$ and $f(b)$ become close. In this case overflow error can occur very easily, and the method fails.

5.4 Newton–Raphson method

The Newton–Raphson method makes explicit use of the derivative of the function of which we wish to find the zero. If we suppose we have more local information about $f(x)$, such as that used in developing a Taylor expansion of the function about the point x_0, we can often find the zeros more quickly. Suppose we have reason to believe that there is a zero of $f(x)$ near the point x_0. The Taylor expansion for $f(x)$ about x_0 can be written as:

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \dots \tag{5.3}$$

If we drop the terms of this expansion beyond the first order term, and write:

$$f(x) \approx f(x_0) + f'(x_0)(x-x_0) \tag{5.4}$$

and set $f(x) = 0$ to find the next approximation x_1 to the zero of $f(x)$, we find:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}\,. \tag{5.5}$$

This provides us with an iteration scheme which may well converge on the zero of $f(x)$, under appropriate conditions.

To examine the conditions under which this iteration converges, we consider the scheme in the approach used previously for fixed point iteration, since we have actually set up a fixed point iteration for the first order Taylor approximation to $f(x)$. The iteration function is:

$$\phi(x) = x - \frac{f(x)}{f'(x)} \tag{5.6}$$

and the derivative is:

$$\phi'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}\,. \tag{5.7}$$

At the actual zero, $f(x) = 0$, so that as long as $f'(x) \neq 0$, we have $g'(x) = 0$ at the zero of $f(x)$. Thus continuity implies that there must be a neighbourhood around the zero where $|g'(x)| < 1$ and we can conclude that the Newton–Raphson method converges in the interval where $\left|f(x)f''(x)/[f'(x)]^2\right| < 1$.

The Newton–Raphson method is illustrated graphically in Figure 5.3 for a simple monotonic function $f(x)$. The iteration from x_0, an initial guess for the zero of $f(x)$, involves drawing the tangent to $f(x)$ from the point $(x_0, f(x0))$ until it intersects the x axis. The point of intersection is the next approximate value, x_1, of the zero of $f(x)$. The process is repeated until the change in the intersection point is smaller than the requested tolerance or precision, i.e. until $|x_n - x_{n-1}| < \epsilon$ for some specified ϵ.

If the restriction that $f'(x) \neq 0$ at or near the zero of $f(x)$ is dropped, the Newton–Raphson method gets more complicated, but can still be applied as follows. Let us suppose that

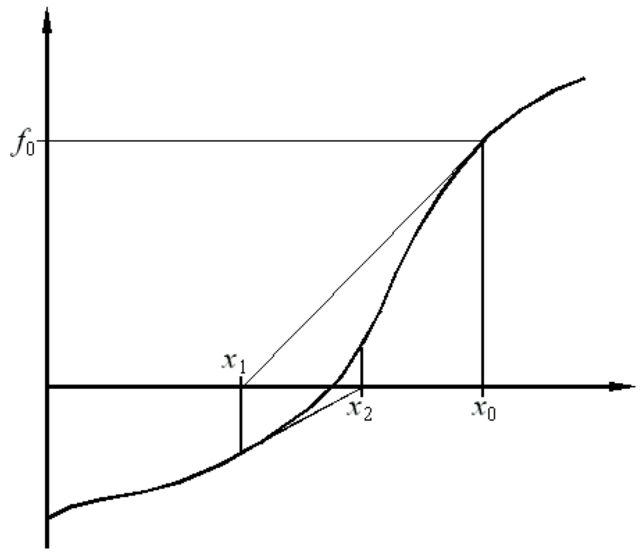

Figure 5.3: Schematic representation of the Newton–Raphson method showing the iteration from the initial guess x0 to the next approximation x1 to the zero of the function $f(x)$.

$f'(a) = 0$ at a zero of $f(x)$ but $f''(a) \neq 0$, so that the Taylor expansion of $f(x)$ about the point $x = a$ becomes:

$$f(x) = \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots \tag{5.8}$$

so that we can rewrite $f(x)$ in the form:

$$f(x) = (x-a)\,h(x) \tag{5.9}$$

where $h(x)$ is a function of the form:

$$h(x) = \frac{f''(a)}{2!}(x-a) + \frac{f'''(a)}{3!}(x-a)^2 + \dots \tag{5.10}$$

and we know that the coefficient of $(x-a)$ does not vanish at $x = a$ since $f''(a) \neq 0$. Then $h(x)$ is a function with a zero at $x = a$, but for which $h'(a) \neq 0$, so that the Newton–Raphson method can be applied to find the zero of $h(x)$ at $x = a$. Once that zero of $h(x)$ is found, it is clear that the problem of $f'(a) = 0$ arises because the zero $x = a$ is a double root of $f(x)$. The same approach applies to even higher order zeroes of $f(x)$. If we find that all the derivatives of $f(x)$ vanish up to $f^{(k)}(x)$ at a point $x = a$, then we should suspect that a zero of multiplicity k+1 is a possibility at $x = a$.

5.4.1 Deflation

The discussion of multiple zeroes of a function $f(x)$ also leads to a technique, called deflation, to handle the difficulties which arise when there are two zeroes which lie so close to each other that once the first is found, it is difficult to prevent the Newton–Raphson iteration from finding it repeatedly, while its nearby companion cannot be found.

If the zero at $x = a$ is a simple zero (i.e. not a multiple zero), then switching consideration from $f(x)$ to $h(x) = f(x)/(x{-}a)$ results in a zero finding problem for $h(x)$, but where we know that $h(x)$ does not have a zero at $x = a$, based on the previous discussion of multiple zeroes. If $f(x)$ really does have a second zero at $x = b$ near to $x = a$, we can be sure that $h(x)$ does not have the zero $x = a$ but does have the zero $x = b$.

The Newton–Raphson method can now be applied to finding the zero of $h(x)$ at $x = b$ without any difficulty from the zero of $f(x)$ at $x = a$. Once the zero of h(x) has been located at $x = b$, we know by the construction of $h(x) = f(x)(x{-}a)$ that $f(x)$ has the same zero at $x = b$.

5.5 The secant method

While the Newton–Raphson method has many positive features, it does require the evaluation of two different functions on each iteration, $f(x)$ and $f'(x)$. When $f(x)$ is reasonably simple, it easy to compute $f'(x)$, but when $f(x)$ is a complicated function, the computation of the derivative can be tedious at best. Thus it is useful to have another method which does not require evaluation of $f'(x)$. Such a method is the secant method which closely resembles the regula falsi method in using a linear approximation to $f(x)$ at each iteration, but with the bracketing aspect dropped.

The secant method assumes a function to be approximately linear in the region of interest. Each improvement is taken as the point where the approximating line crosses the axis. The secant method retains only the most recent estimate, so the root does not necessarily remain bracketed.

This method is a variant of the regula falsi, except that there is no bracketing requirement exists, and the linear interpolation becomes linear extrapolation.

In Fig. 5.4 the secant method is illustrated and we can see that the function $f(x)$ is being approximated by a straight line which is an extrapolation based on the two points x_0 and x_1. The line passing through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ can be seen to be given by:

$$y - f_1 = (x - x_1)\frac{f_1 - f_0}{x_1 - x_0}\ , \tag{5.11}$$

so that solving for the value of x for which $y = 0$ for this line, we have the two-point iteration formula:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\ . \tag{5.12}$$

The secant method is closely related to the Newton–Raphson method, and this relationship is clearly related to the fact that the quantity

$$\frac{\Delta f}{\Delta x} = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \tag{5.13}$$

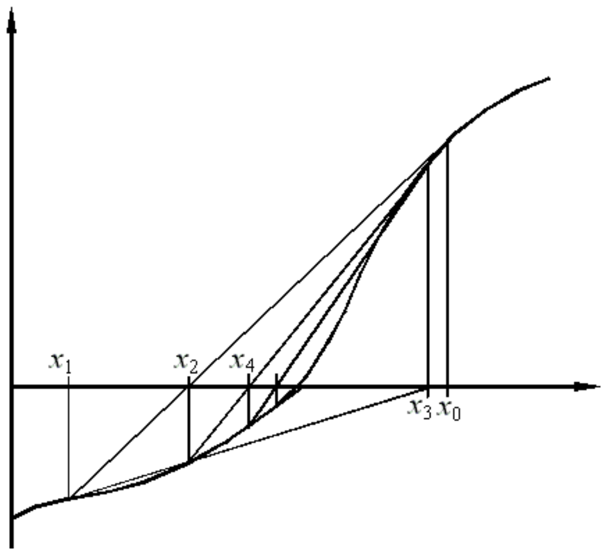

Figure 5.4: The secant method

becomes $f'(x)$ in the limit that $|x_k - x_{k-1}| \to 0$. In fact, from the *mean value theorem* of calculus, we know that as long as $f(x)$ is continuous in the interval $[x_k, x_{k-1}]$, there is a point x = c in that interval for which

$$\frac{\Delta f}{\Delta x} = f'(c)\,. \qquad (5.14)$$

The secant method does not require the evaluation of any formal derivative as the Newton–Raphson does, and requires only one evaluation of $f(x)$ per iteration. In many cases the absence of any requirement to compute a derivative is a significant advantage, not only because there is no need to perform the formal differentiation, but because frequently the derivative of a function is a significantly more complicated function than the original function.

5.6 Convergence

5.6.1 Bisection and regula falsi

We cannot expect to find exact solutions s_* of an equation $f(s) = 0$. Rather, the aim is to find value x_0 which satisfies

$$|x_0 - s_*| < \epsilon\,, \qquad (5.15)$$

i.e. which differs from the exact solution s_* by less than a given error tolerance ϵ.

The bisection method is an almost foolproof method, although it may converge slowly towards the solution due to the fact that it halves the intervals. After n divisions by 2 we have a possible solution in the interval with length

$$\frac{1}{2^n}|b-a|\,,$$

and if we set $x_0 = (a+b)/2$ and let x_n be the midpoints in the intervals, we find after n iterations

$$|s-x_n| \leq \frac{1}{2^{n+1}}|b-a| \, , \tag{5.16}$$

since the n-th interval has length $|b-a|/2^n$. Note that this convergence criterion is independent of the actual function $f(x)$ as long as this function fulfils the condition discussed above.

Basically the smallest interval is limited by the machine accuracy ϵ, so that

$$\frac{1}{2^{n+1}}|b-a| \; = \epsilon \, , \tag{5.17}$$

Hence if we know that a root lies between 0.5 and 1, and we are using floating point arithmetic with a 24 bit mantissa, the interval can be halved at most 23 times.

As for the *regula falsi*, there is no strict precision guarantee. One cannot predict the number of iterations to reach a given precision!

5.6.2 Newton–Raphson and secant methods

The rate of convergence for the Newton–Raphson for a simple zero $x = a$ is said to be of order 2, meaning that the error $|a - x_{k+1}|$ at iteration k+1 is related to the error $|a - x_k|$ at iteration k by the relation:

$$|a-x_{k+1}| \approx A|a-x_k|^R \tag{5.18}$$

where A is some constant factor, and $R = 2$. Thus if the error $|a - x_k| \approx 10^{-3}$, then the error in the next iteration will be $|a - x_{k+1}| \approx A \times 10^{-6}$. The exact value of A is not particularly important here, for the exponent R has the dominant effect. While $R = 2$ for the Newton–Raphson method, the value for the secant method is about 1.618 [actually it can be shown to be $(1+\sqrt{5})/2$] for simple zeroes. For many practical purposes, the difference between convergence of order 2 and order 1.618 is of less impact than the difficulties already mentioned for the Newton–Raphson method.

While both the regula falsi and secant methods use the idea of a linear approximation to the function based on its values at two points, the regula falsi method depends on the fact that those two points enclose a zero, with the consequent sign change for $f(x)$, while the secant method simply extrapolates using these two points to find the next approximation to the zero. The regula falsi method is sure to find the zero, since it keeps it bracketed, while the secant method can sometimes fail to find a zero that does exist. The secant method has the advantage that we do not need to have prior knowledge of the interval in which each zero of $f(x)$ lies.

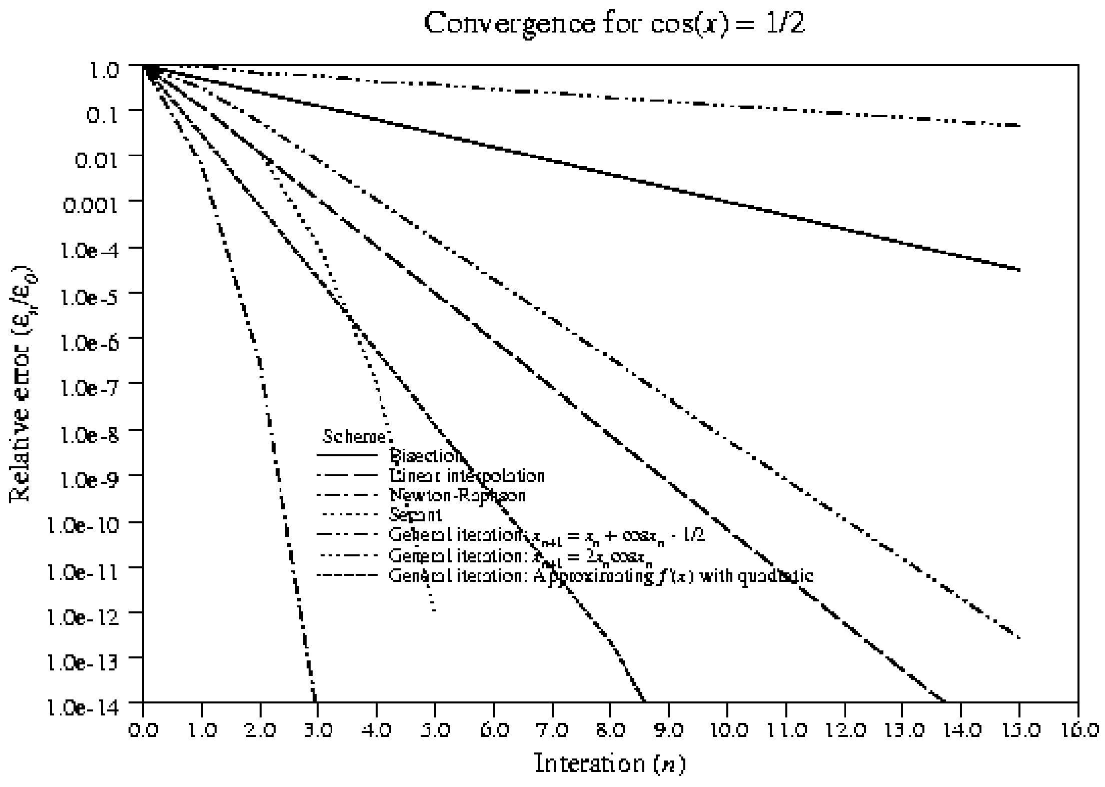

Figure 5.5: Comparison of the convergence of the error in the estimate of the root of $\cos x = 1/2$.

5.6.3 Example

Figure 5.5 compares of the convergence of the error in the estimate of the root to $\cos x = 1/2$ for a range of different root finding algorithms. The clear winner among the methods we covered in the previous sections is the Newton–Raphson scheme. However, this changes from one function to another and it is up to you to build the algorithm that selects the most adequate method.

5.7 Appendix

5.7.1 The '`FindRoot`' command in Mathematica

'`FindRoot`' uses the Newton–Raphson method and the secant method.

Note: The Newton method is implemented in Mathematica as the undocumented option *`Method→Newton`* in '`FindRoot[eqn, {`x, x_0, x_1`}]`'.

The secant method is implemented in Mathematica as the undocumented option *`Method→Secant`* in '`FindRoot[eqn, {`x, x_0, x_1`}]`'.

Note: Advanced Root finding tools make use of more than one method to find the solution. In many cases, bisection or the regula falsi is used to get close enough to the solution as to switch to the Newton–Raphson or the secant method.

The bisection method and regula falsi are not implemented in Mathematica, so here is how one can write the corresponding notebooks files (these can be downloaded from `http://www.capca.ucalgary.ca/top/teaching/phys499+535/Section5`):

Bisection

```
Bisection[a0_, b0_, m_] :=
    Module[{a = N[a0], b = N[b0]},
        c = 0.5*(a+b);
        k = 0;
        While[ k < m,
          If[Sign[f[b]] == Sign[f[c]], b = c, a = c;];
          c = 0.5*(a+b)
          k = k + 1;
          output = Append[output, {k,a,c,b,f[c]}];
        ];
        Print[NumberForm[TableForm[output,
            TableHeadings → {None,{"k","a_k",
            "c_k","b_k","f[c_k]"}}],16]];
        Print[" c = ", NumberForm[c,16]];
        Print[" δc = ±", 0.5*(b-a)];
        Print["f[c] = ", NumberForm[f[c],16]];
    ]
```

Regula falsi

```
RegulaFalsi[a0_, b0_, m_] :=
Module[{}
    a = N[a0];
    b = N[b0];
    c = (a f[b] - b f[a]) / (f[b] - f[a]);
    k = 0;
    output = {{k,a,c,b,f[c]}};
    While[ k < m,
    If[Sign[f[b]] == Sign[f[c]],b = c,a = c;];
    c = (a f[b] - b f[a]) / (f[b]-f[a]);
    k = k + 1;
    output = Append[output, {k,a,c,b,f[c]}];];
    Print[NumberForm[TableForm[output,
          TableHeadings→ {None,{"k","a_k",
          "c_k","b_k","f[c_k]"}}],16]];
      Print[" c = ", NumberForm[c,16]];
      Print[" δc = ±", 0.5*(b-a)];
      Print["f[c] = ", NumberForm[f[c],16]];
  ]
```

Application

Use the modules given above to solve $x^3 + 4x^2 - 10 = 0$ in the interval $[0, 1]$. How many iterations are required for convergence?