Numerical methods and optimization

Week 1.

Introduction

In mathematics and computing, a root-finding algorithm is an algorithm for finding zeroes, also called "roots", of continuous functions. A zero of a function f, from the real numbers to real numbers is a number x such that

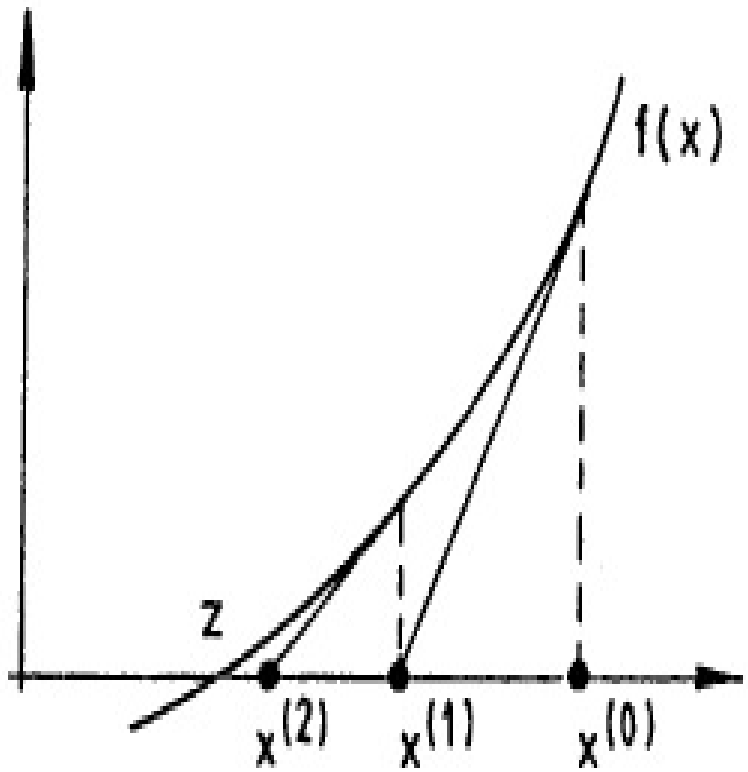
$$f(x) = 0.$$

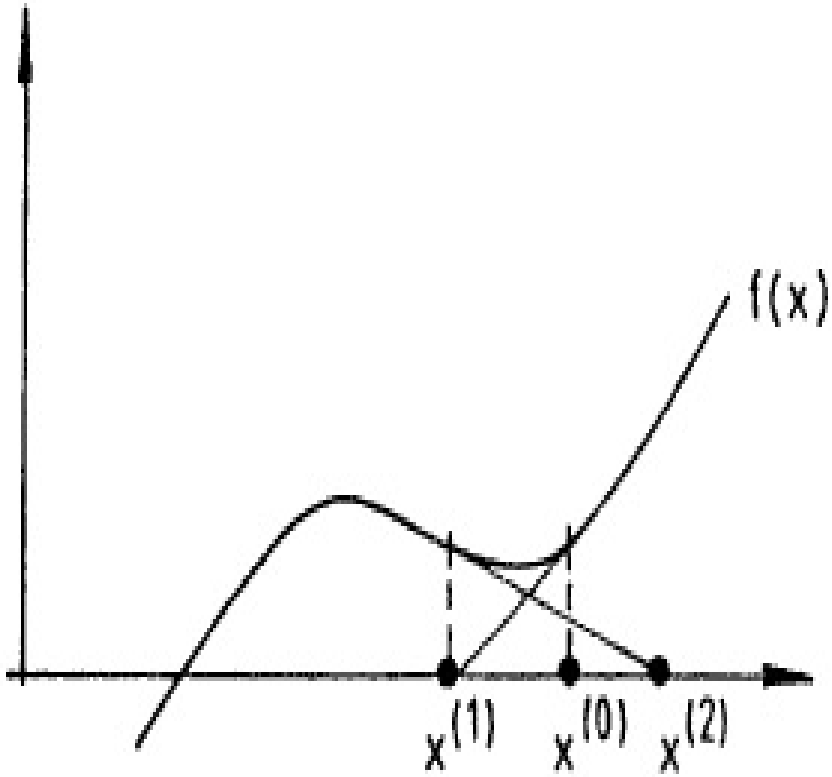
As, generally, the zeroes of a function cannot be computed exactly nor expressed in closed form, root-finding algorithms provide approximations to zeroes, expressed either as floating point numbers or as small isolating intervals, or disks for complex roots (an interval or disk output being equivalent to an approximate output together with an error bound).

Solving an equation $f(x) = g(x)$ is the same as finding the roots of the function

$$h(x) = f(x) - g(x).$$

Thus root-finding algorithms allow solving any equation defined by continuous functions.
However, most root-finding algorithms do not guarantee that they will find all the roots; in particular, if such an algorithm does not find any root, that does not mean that no root exists.
Most numerical root-finding methods use iteration, producing a sequence of numbers that hopefully converge towards the root as a limit.


f(x)
z
x(2)
x(1)
x(0)


f(x)
x(1)
x(0)
x(2)

They require one or more initial guesses of the root as starting values, then each iteration of the algorithm produces a successively more accurate approximation to the root.
Since the iteration must be stopped at some point these methods produce an approximation to the root, not an exact solution.
Many methods compute subsequent values by evaluating an auxiliary function on the preceding values. The limit is thus a fixed point of the auxiliary function, which is chosen for having the roots of the original equation as fixed points, and for converging rapidly to these fixed points.

Bracketing Method

Bracketing methods determine successively smaller intervals (brackets) that contain a root.
When the interval is small enough, then a root has been found.
They generally use the intermediate value theorem, which asserts that if a continuous function has values of opposite signs at the end points of an interval, then the function has at least one root in the interval.
Therefore, they require to start with an interval such that the function takes opposite signs at the end points of the interval.
However, in the case of polynomials there are other methods for getting information on the number of roots in an interval. They lead to efficient algorithms for real-root isolation of polynomials, which ensure finding all real roots with a guaranteed accuracy.

Bisection method

The simplest root-finding algorithm is the **bisection method**. Let f be a continuous function, for which one knows an interval $[a, b]$ such that $f(a)$ and $f(b)$ have opposite signs (a bracket). Let

$$c = (a + b)/2$$

be the middle of the interval (the midpoint or the point that bisects the interval). Then either $f(a)$ and $f(c)$, or $f(c)$ and $f(b)$ have opposite signs, and one has divided by two the size of the interval. (If $f(c) = 0$ then c may be taken as the solution and the process stops.)

The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

Remark: Although the bisection method is robust, it gains one and only one bit of accuracy with each iteration. Other methods, under appropriate conditions, can gain accuracy faster.

The algorithm of Bisection method

1 INPUT: Function f, endpoint values a, b, tolerance TOL, maximum iterations $NMAX$.
2 CONDITIONS: $a < b$, either $f(a) < 0$ and $f(b) > 0$ or $f(a) > 0$ and $f(b) < 0$.
3 OUTPUT: value which differs from a root of $f(x) = 0$ by less than TOL.
4 $N \leftarrow 1$
5 WHILE $N \leq NMAX$ do // limit iterations to prevent infinite loop
6 $c \leftarrow (a + b)/2$ // new midpoint
7 if $f(c) = 0$ or $(b - a)/2 < TOL$ then // solution found
8 THEN Output(c) and STOP
9 $N \leftarrow N + 1$
10 if $sign(f(c)) = sign(f(a))$ then $a \leftarrow c$ else $b \leftarrow c$ // new inte
11 Output("Method failed.") // max number of steps exceeded

Example

Suppose that the bisection method is used to find a root of the polynomial

$$f(x) = x^3 - x - 2$$

Example

Suppose that the bisection method is used to find a root of the polynomial

$$f(x) = x^3 - x - 2$$

Solution

Step 1. First, two numbers a and b have to be found such that $f(a)$ and $f(b)$ have opposite signs. For the above function, $a = 1$ and $b = 2$ satisfy this criterion, as

$$f(1) = (1)^3 - (1) - 2 = -2$$

and

$$f(2) = (2)^3 - (2) - 2 = +4$$

Solution

Because the function is continuous, there must be a root within the interval $[1, 2]$.
In the first iteration, the end points of the interval which brackets the root are $a_1 = 1$ and $b_1 = 2$, so the midpoint is

$$c_1 = \frac{2+1}{2} = 1.5$$

The function value at the midpoint is $f(c_1) = (1.5)^3 - (1.5) - 2 = -0.125$. Because $f(c_1)$ is negative, $a_1 = 1$ is replaced with $a_2 = 1.5$ (and $b_2 = 2$) for the next iteration to ensure that $f(a)$ and $f(b)$ have opposite signs. As this continues, the interval between a and b will become increasingly smaller, converging on the root of the function. See this happen in the table below.

Solution

Iteration	a_n	b_n	c_n	$f(c_n)$
1	1	2	1.5	−0.125
2	1.5	2	1.75	1.6093750
3	1.5	1.75	1.625	0.6660156
4	1.5	1.625	1.5625	0.2521973
5	1.5	1.5625	1.5312500	0.0591125
6	1.5	1.5312500	1.5156250	−0.0340538
7	1.5156250	1.5312500	1.5234375	0.0122504
8	1.5156250	1.5234375	1.5195313	−0.0109712
9	1.5195313	1.5234375	1.5214844	0.0006222
10	1.5195313	1.5214844	1.5205078	−0.0051789
11	1.5205078	1.5214844	1.5209961	−0.0022794
12	1.5209961	1.5214844	1.5212402	−0.0008289
13	1.5212402	1.5214844	1.5213623	−0.0001034
14	1.5213623	1.5214844	1.5214233	0.0002594
15	1.5213623	1.5214233	1.5213928	0.0000780

The absolute error is halved at each step so the method converges linearly.
Specifically, if $c_1 = (a+b)/2$ is the midpoint of the initial interval, and c_n is the midpoint of the interval in the nth step, then the difference between c_n and a solution c is bounded by

$$|c_n - c| \leq \frac{|b-a|}{2^n}.$$

This formula can be used to determine, in advance, an upper bound on the number of iterations that the bisection method needs to converge to a root to within a certain tolerance. The number n of iterations needed to achieve a required tolerance TOL (that is, an error guaranteed to be at most TOL), is bounded by

$$n \leq NMAX \equiv \left\lceil \log_2 \left(\frac{L_0}{TOL} \right) \right\rceil,$$

where the initial bracket size $L_0 = |b - a|$ and the required bracket size $TOL \leq L_0$.
The main motivation to use the bisection method is that over the set of continuous functions, no other method can guarantee to produce an estimate c_n to the solution c that in the worst case has an TOL absolute error with less than $NMAX$ iterations. This is also true under several common assumptions on function f and the behaviour of the function in the neighbourhood of the root.