Rate of convergence and bisection

ASSIGN BUSTER

Rate of convergence estimate of the speed with which a given sequence or iteration approaches its limit, often measured by the number of terms or evaluations involved in obtaining a given accuracy. Although strictly speaking, a limit does not give information about any finite first part of the sequence, this concept is of practical importance if we deal with a sequence of successive approximations for an iterative method, as then typically fewer iterations are needed to yield a useful approximation if the rate of convergence is higher. This may even make the difference between needing ten or a million iterations.

Rate of convergence is measured in terms of rate at which the relative error decreases between successive approximations. There are mainly two type of convergence: linear and quadratic.

Convergence of a sequence subject to the condition, for p > 1, that

as n increases is called pth-order convergence; for example, quadratic convergence when p = 2. One similarly speaks of logarithmic convergence or exponential convergence.

The Bisection Method

In mathematics, the bisection method is a root-finding algorithm which repeatedly bisects an interval then selects a subinterval in which a root must lie for further processing. It is a very simple and robust method, but it is also relatively slow. The bisection method is simple, robust, and straight-forward: take an interval [a, b] such that f(a) and f(b) have opposite signs, find the midpoint of [a, b], and then decide whether the root lies on [a, (a + b)/2] or [(a + b)/2, b]. Repeat until the interval is sufficiently small.

The bisection method, suitable for implementation on a computer allows to find the roots of the equation f (x) = 0, based on the following theorem:

Theorem: If f is continuous for x between a and b and if f (a) and f(b) have opposite signs, then there exists at least one real root of f (x) = 0 between a and b.

Procedure: Suppose that a continuous function f is negative at x = a and positive at x = b, so that there is at least one real root between a and b. (As a rule, a and b may be found from a graph of f.) If we calculate f ((a +b)/2), which is the function value at the point of bisection of the interval a < x < b, there are three possibilities:

f ((a + b)/2) = 0, in which case (a + b)/2 is the root;

f ((a + b)/2) <0, in which case the root lies between (a + b)/2 and b;

f ((a + b)/2) > 0, in which case the root lies between a and (a + b)/2.

Advantages and drawbacks of the bisection method

Advantages of Bisection Method

The bisection method is always convergent. Since the method brackets the root, the method is guaranteed to converge.

As iterations are conducted, the interval gets halved. So one can guarantee the decrease in the error in the solution of the equation.

Drawbacks of Bisection Method

The convergence of bisection method is slow as it is simply based on halving the interval.

If one of the initial guesses is closer to the root, it will take larger number of iterations to reach the root.

If a function is such that it just touches the x-axis (Figure 3. 8) such as

it will be unable to find the lower guess, , and upper guess, , such that

For functions where there is a singularity and it reverses sign at the singularity, bisection method may converge on the singularity (Figure 3. 9).

An example include

and, are valid initial guesses which satisfy

•

However, the function is not continuous and the theorem that a root exists is also not applicable.

Figure. 3. 8. Function has a single root at that cannot be bracketed.

Figure. 3. 9. Function has no root but changes sign.

False position method

The false-position method is a modification on the bisection method. The false position method or regula falsi method is a root-finding algorithm that combines features from the bisection method and the secant method. If it is known that the root lies on [a, b], then it is reasonable that we can approximate the function on the interval by interpolating the points (a, f(a)) and (b, f(b)). The method of false position dates back to the ancient Egyptians. It remains an effective alternative to the bisection method for

solving the equation f(x) = 0 for a real root between a and b, given that f (x) is continuous and f (a) and f(b) have opposite signs. The algorithm is suitable for automatic computation

Procedure:

The curve y = f(x) is not generally a straight line. However, one may join the points (a, f(a)) and (b, f(b)) by the straight line

Thus straight line cuts the x-axis at (X, 0) where

so that

Suppose that f(a) is negative and f(b) is positive. As in the bisection method, there are the three possibilities :

f(X) = 0, when case X is the root ;

f(X) < 0, when the root lies between X and b ;

f(X)> 0, when the root lies between X and a.

Again, in Case 1, the process is terminated, in either Case 2 or Case 3, the process can be repeated until the root is obtained to the desired accuracy.

Convergence of False Position Method and Bisection Method

Source code for False Position Method:

Example code of False-position method

C code was written for clarity instead of efficiency. It was designed to solve the same problem as solved by the Newton’s method and secant method

code: to find the positive number x where cos(x) = x3. This problem is transformed into a root-finding problem of the form

f(x) = cos(x) – x3 = 0.

```
#include

#include

double f(double x)

{
return cos(x) – x*x*x;

}
double FalsiMethod(double s, double t, double e, int m)

{
int n, side= 0;

double r, fr, fs = f(s), ft = f(t);

for (n = 1; n <= m; n++)

{
r = (fs*t – ft*s) / (fs – ft);

if (fabs(t-s) < e*fabs(t+s)) break;

fr = f(r);

if (fr * ft > 0)
```

```
{
t = r; ft = fr;

if (side==-1) fs /= 2;

side = -1;

}
else if (fs * fr > 0)

{
s = r; fs = fr;

if (side==+1) ft /= 2;

side = +1;

}
else

break;

}
return r;

}
int main(void)

{
printf("%0. 15fn", FalsiMethod(0, 1, 5E-15, 100));

return 0;
```

}

After running this code, the final answer is approximately 0.

865474033101614

Example 1

Consider finding the root of f(x) = x2 – 3. Let Îµstep = 0. 01, Îµabs = 0. 01 and start with the interval [1, 2].

Table 1. False-position method applied to f(x) = x2 – 3.

a	b	f(a)	f(b)	c	f(c)	Update	Step Size
1. 0	2. 0	-2. 00	1. 00	1. 6667	-0. 2221		

a = c

0. 6667

1. 6667

2. 0

-0. 2221

1. 0

1. 7273

-0. 0164

a = c

0. 0606

1. 7273

2. 0

-0. 0164

1. 0

1. 7317

0. 0012

a = c

0. 0044

Thus, with the third iteration, we note that the last step 1. 7273 â†’ 1. 7317 is less than 0. 01 and | f(1. 7317)| < 0. 01, and therefore we chose b = 1. 7317 to be our approximation of the root.

Note that after three iterations of the false-position method, we have an acceptable answer (1. 7317 where f(1. 7317) = -0. 0044) whereas with the bisection method, it took seven iterations to find a (notable less accurate) acceptable answer (1. 71344 where f(1. 73144) = 0. 0082)

Example 2

Consider finding the root of f(x) = e-x(3. 2 sin(x) – 0. 5 cos(x)) on the interval [3, 4], this time with Îµstep = 0. 001, Îµabs = 0. 001.

Table 2. False-position method applied to f(x) = e-x(3. 2 sin(x) – 0. 5 cos(x)).

a

b

f(a)

f(b)

c

f(c)

Update

Step Size

3. 0

4. 0

0. 047127

-0. 038372

3. 5513

-0. 023411

b = c

0. 4487

3. 0

3. 5513

0. 047127

-0. 023411

3. 3683

-0. 0079940

b = c

0. 1830

3. 0

3. 3683

0. 047127

-0. 0079940

3. 3149

-0. 0021548

b = c

0. 0534

3. 0

3. 3149

0. 047127

-0. 0021548

3. 3010

-0. 00052616

b = c

0. 0139

3. 0

3. 3010

0. 047127

-0. 00052616

3. 2978

-0. 00014453

b = c

0. 0032

3. 0

3. 2978

0. 047127

-0. 00014453

3. 2969

-0. 000036998

b = c

0. 0009

Thus, after the sixth iteration, we note that the final step, 3. 2978 â†’ 3. 2969 has a size less than 0. 001 and | f(3. 2969)| < 0. 001 and therefore we chose b = 3. 2969 to be our approximation of the root.

In this case, the solution we found was not as good as the solution we found using the bisection method (f(3. 2963) = 0. 000034799) however, we only used six instead of eleven iterations.

Source code for Bisection method

```
#include

#include

#define epsilon 1e-6

main()

{
double g1, g2, g, v, v1, v2, dx;

int found, converged, i;

found= 0;

printf(" enter the first guessn");

scanf("%lf",&g1);

v1= g1*g1*g1-15;

printf(" value 1 is %lfn", v1);

while (found== 0)

{
printf(" enter the second guessn");

scanf("%lf",&g2);

v2= g2*g2*g2-15;

printf(" value 2 is %lfn", v2);
```

```
if (v1*v2> 0)

{found= 0;}

else

found= 1;

}
printf(“ right guessn”);

i= 1;

while (converged== 0)

{
printf(“ n iteration=%dn”, i);

g=(g1+g2)/2;

printf(“ new guess is %lfn”, g);

v= g*g*g-15;

printf(“ new value is%lfn”, v);

if (v*v1> 0)

{
g1= g;

printf(“ the next guess is %lfn”, g);
```

```
dx=(g1-g2)/g1;

}
else

{
g2= g;

printf(“ the next guess is %lfn”, g);

dx=(g1-g2)/g1;

}
if (fabs(dx)’less than’ epsilon

{converged= 1;}

i= i+1;

}
printf(“ nth calculated value is %lfn”, v);

}
```

Example 1

Consider finding the root of f(x) = x2 – 3. Let Îµstep = 0. 01, Îµabs = 0. 01 and start with the interval [1, 2].

Table 1. Bisection method applied to f(x) = x2 – 3.

a

b

f(a)

f(b)

c = (a + b)/2

f(c)

Update

new b âˆ’ a

1. 0

2. 0

-2. 0

1. 0

1. 5

-0. 75

a = c

0. 5

1. 5

2. 0

-0. 75

1. 0

1. 75

0. 062

b = c

0. 25

1. 5

1. 75

-0. 75

0. 0625

1. 625

-0. 359

a = c

0. 125

1. 625

1. 75

-0. 3594

0. 0625

1. 6875

-0. 1523

a = c

0. 0625

1. 6875

1. 75

-0. 1523

0. 0625

1. 7188

-0. 0457

a = c

0. 0313

1. 7188

1. 75

-0. 0457

0. 0625

1. 7344

0. 0081

b = c

0. 0156

1. 71988

1. 7344

-0. 0457

0. 0081

1. 7266

-0. 0189

a = c

0. 0078

Thus, with the seventh iteration, we note that the final interval, [1. 7266, 1. 7344], has a width less than 0. 01 and | f(1. 7344)| < 0. 01, and therefore we chose b = 1. 7344 to be our approximation of the root.

Example 2

Consider finding the root of f(x) = e-x(3. 2 sin(x) – 0. 5 cos(x)) on the interval [3, 4], this time with Îµstep = 0. 001, Îµabs = 0. 001.

Table 1. Bisection method applied to f(x) = e-x(3. 2 sin(x) – 0. 5 cos(x)).

a

b

f(a)

f(b)

c = (a + b)/2

f(c)

Update

new b âˆ’ a

3. 0

4. 0

0. 047127

-0. 038372

3. 5

-0. 019757

b = c

0. 5

3. 0

3. 5

0. 047127

-0. 019757

3. 25

0. 0058479

a = c

0. 25

3. 25

3. 5

0. 0058479

-0. 019757

3. 375

-0. 0086808

b = c

0. 125

3. 25

3. 375

0. 0058479

-0. 0086808

3. 3125

-0. 0018773

b = c

0. 0625

3. 25

3. 3125

0. 0058479

-0. 0018773

3. 2812

0. 0018739

a = c

0. 0313

3. 2812

3. 3125

0. 0018739

-0. 0018773

3. 2968

-0. 000024791

b = c

0. 0156

3. 2812

3. 2968

0. 0018739

-0. 000024791

3. 289

0. 00091736

a = c

0. 0078

3. 289

3. 2968

0. 00091736

-0. 000024791

3. 2929

0. 00044352

a = c

0. 0039

3. 2929

3. 2968

0. 00044352

-0. 000024791

3. 2948

0. 00021466

a = c

0. 002

3. 2948

3. 2968

0. 00021466

-0. 000024791

3. 2958

0. 000094077

a = c

0. 001

3. 2958

3. 2968

0. 000094077

-0. 000024791

3. 2963

0. 000034799

a = c

0. 0005

Thus, after the 11th iteration, we note that the final interval, [3. 2958, 3. 2968] has a width less than 0. 001 and | f(3. 2968)| < 0. 001 and therefore we chose b = 3. 2968 to be our approximation of the root.

Comparison of rate of convergence for bisection and false-position method

Like the bisection method, the method of false position has almost assured convergence, and it may converge to a root faster. Finally, note that bisection is rather slow; after n iterations the interval containing the root is of length (b – a)/2n. However, provided values of f can be generated readily, as when a computer is used, the rather large number of iterations which can be involved in the application of bisection is of relatively little consequence.

The false position method would be better i. e. converges to the root more rapidly as it takes into account the relative magnitudes of f(b) and f(a) unlike bisection which just uses the midpoint of a and b, where [a, b] is the interval over which the root occurs. Following is the example of the convergence rate of bisection method and false position method for the similar equation which shows that rate of convergence of false position method is faster than that of the bisection method.