**Bisection Method**

**Bisection Method**

二分法是一种查找方程零值点的方法，它反复地对一个区间均分，直到找到满足要求的零值点。二分法简单，但是收敛速度较慢，为线性收敛。

设有函数在区间上连续，并且与异号，则在区间上至少有一个解。在迭代的每一步中，都会计算中间点，将原始区间分为两个子区间和。在第一次迭代中，如果刚好，则点就是在区间上的零值点（不过这一般是不太可能的）；如果，就只有下面的两种情况了：

1. 如果与同号，则新的迭代区间变为，并且将赋值于

2. 如果与反号，则新的迭代区间变为，并且将赋值于

直到达到计算的要求：提前设置精度要求eps，当区间的绝对误差小于eps时，结束计算。

这里给出提前确定所需精度后（精确到小数点后p位），估计计算迭代次数的公式



**伪代码**

INPUT: 函数、区间，精度eps

if f(a)\*f(b) > 0

print\*, " error: f(a) \* f(b) < 0 is not satisfied! "

Do while ( b-a ) > eps

c ← (a + b) / 2

get f(c)

if f(c)==0 stop end

if f(a)\*f(c) > 0 then

a ← c; f(a) ← f(c)

else

b ← c; f(b) ← f(c)

end if

end do

**实例**

设函数在区间上有一零值点，求其零值点。

**源代码**

**Fortran**

Program BisectionMethod

!// 二分法求解方程f(t) = cos(t) - t在区间[0,1]上的根

Implicit none

Real(kind=8), parameter :: eps = 1d-12 !// 精度控制

Real(kind=8) :: a = 0.d0, b = 1.d0, c, t

Real(kind=8) :: fa, fb, fc

fa = cos(a) - a !// 方程为: f(x) = cos(x) - x

fb = cos(b) - b

If ( fa \* fb > 0.d0 ) then

print\*, 'error: f(a)\*f(b) < 0 not satisfied!'

stop

end if

Do while ( (b - a) > eps )

c = ( b + a ) / 2.d0

fc = cos(c) - c

If ( fc == 0.d0 ) exit

If ( fa \* fc > 0.d0 ) then

a = c; fa = fc

else

b = c ; fb = fc

End if

End do

t = ( b + a ) / 2.d0

Write (\*,'(1x,A,f20.15)') 't =', t

Write (\*,'(" cos(t) - t =",f20.15)') cos(t) - t

End program BisectionMethod

**Python**

|  |  |
| --- | --- |
|  | """ |
|  | Created on Fri Apr 13 14:34:13 2018 |
|  |  |
|  | @author: luk |
|  | """ |
|  |  |
|  | # 二分法求解方程f(t) = cos(t) - t 在区间[0,1]上的解 |
|  | eps = 1e-12; # 控制精度 |
|  | a = 0.0; b = 1.0; # 搜索区间 |
|  |  |
|  | import math |
|  | fa = math.cos( a ) - a; |
|  | fb = math.cos( b ) - b; |
|  | if fa \* fb > 0.0: |
|  | print ( " error: f(a) \* f(b) < 0 is not satisfied! " ) |
|  | input() |
|  | quit() |
|  |  |
|  |  |
|  | while ( b - a ) > eps: |
|  | c = ( a + b ) / 2.0 |
|  | fc = math.cos( c ) - c |
|  | if fc == 0.0: |
|  | break |
|  | if fa \* fc > 0.0: |
|  | a = c; fa = fc; |
|  | elif fa \* fc < 0.0: |
|  | b = c; fb = fc; |
|  |  |
|  | t = ( b + a ) / 2.0 |
|  | print ( " solution is ", t ) |
|  | print ( " error is ", abs(math.cos(t) - t) ) |
|  | input() |

运行结果如下

