**Computational Finance Homework 3 Rootfinding**

**Name: Jian Wang FSUID: Jw09r**

**0. Executive Summary**

Use the Bisect and Newton method and combine those two methods together to find the effective annual interest rate if $1000 per month, compounded monthly and invested over a 15 year period, grows to $1 million at the end of t that period.

My result is that the compounded interest rate is .

1. **Statement of Problem**

The accumulated savings in an account into which regular periodic payments are made is given by:

Where A is the amount in the account. P is the deposit amount, and r is the interest rate per period. The main purpose for me is to find the effective annual interest rate. I firstly use the bisect method to find the root in one digits and then use the Newton method to find the result in the **IEEE single precision machine accuracy**.

1. **Description of The Mathematics**
2. Bisection method:

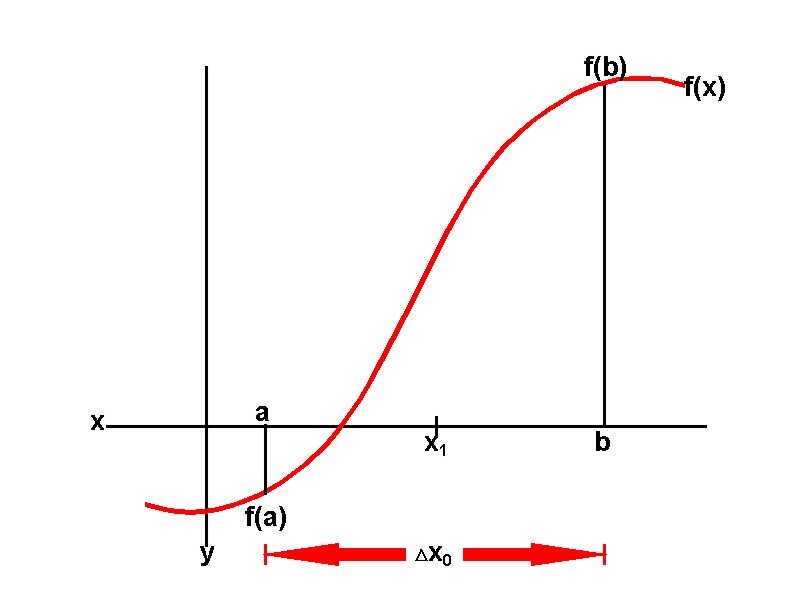
The bisection method requires two initial points a and b such that f(a) and f(b) have opposite signs.

Firstly, choose two ending points that the function values at these two points have opposite signs.

Secondly, then divides the interval in two by computing the midpoint p = (a+b) / 2 of the interval.

Thirdly, if F(a)\*F(P) >0 then a=p else b=p

We can see the chart as follows:



The properties of the convergence is as follows:

We can see that



When then we have Abs error test

then we have the Rel error test.

The Tol should be:

Rel Tol 

Abs Tol smallest distance.

In addition, since, we may choose  as the upper bound of error. So the final criterion to check if it is converge is:



At last, we may calculate the maximum round number when it converges given two starting points, absolute tolerance and relative tolerance.

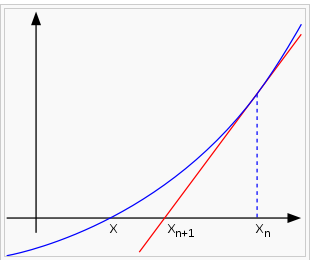


1. Newton’s method:

Find roots of successive linear approximations of F start with the initial point

follow the irritation until it converges. 

it can be seen clearly from the following chart.



From the Taylor expansion:

, where =p, =0, choose x=, then: .





We can see that Newton’s method can converge more quickly than the Bisection method.

Besides, we can also use the criterion as in the bisection method:



1. Description of the Algorithm
2. The Algorithm of the Bisection Method:

For i = 1 to N Do

P=

If converged Exit

If F(a)\*F(P)>0 then

a=P

Else

b=P

next i

Here the convergence is:



You can see the interest rate should be between [0,1]

So I choose the first interval be [0,1]

1. The algorithm of the Newton Method

Given P=

For i = 1 to N Do

P=P-F(P)/F’(P)

If converged Exit

Next i

Here the convergence is as follows:



Since we choose the IEEE single Precision machine accuracy and chose the 1 significant digit in the Bisection method.

**So for the Bisection Method:**

The Absolute Tolerance is

And the related tolerance is 0.5\* which is 0.05

**For the Newton method:**

The Absolute Tolerance is and the related tolerance is 0.5\*

The formula of the Total amount is

So the derivate is that

1. Results

First use bisection method, with ending points 0.001 and 0.1, and after six rounds to find a root, 1.56953%, within 1 effective digital. Then use this value as a start-value to run Newton method, and after 3 rounds I get a final result, 1.57967%, to IEEE single precision machine accuracy. Then using the formula  to get effective annual return rate: 20.6928%

1. Conclusions

From mathematics listed above, we usually use del,, to approximate  which is 8.81088e-8 less than single precision machine epsilon. Also the del,, decrease nearly quadratically, as the mathematics tells me.

So it is reasonable to expect the Newton method converge and the root found is right.

1. Code Listing

<rootfind.h>

#define ROOTFINDING\_H

#include "FMFLOAT.h"

#define FMFLOAT float

class rootfinding

{

public:

FMFLOAT Bisect(FMFLOAT (\*f1)(FMFLOAT x), FMFLOAT a, FMFLOAT b, FMFLOAT absTol, FMFLOAT relTol);

FMFLOAT Newton(FMFLOAT (\*f1)(FMFLOAT x), FMFLOAT (\*f2)(FMFLOAT y), FMFLOAT startvalue, FMFLOAT, FMFLOAT);

FMFLOAT Solve(FMFLOAT (\*f1)(FMFLOAT), FMFLOAT (\*f2)(FMFLOAT), FMFLOAT a, FMFLOAT b, FMFLOAT absTol, FMFLOAT relTol);

private:

int sign(FMFLOAT (\*f)(FMFLOAT), FMFLOAT i);

};

<rootfinding.cpp>

#include "rootfinding.h"

#include <math.h>

#include <float.h>

#include <iostream>

using namespace std;

int rootfinding::sign(FMFLOAT (\*f)(FMFLOAT),FMFLOAT i)

{

if (f(i)>0) return 1;

else return -1;

}

FMFLOAT rootfinding::Bisect(FMFLOAT (\*f1)(FMFLOAT x), FMFLOAT a, FMFLOAT b, FMFLOAT absTol, FMFLOAT relTol)

{

if (sign(f1,a)\*sign(f1,b)>0)

{

cout<<"please choose ending points with opposite sign"<<endl;

return 0;

}

else

{

cout<<"Bisection function: "<<endl;

cout<<"begining point: "<<a<<endl<<"ending point: "<<b<<endl;

cout<<"absolute tolerance: "<<absTol<<endl<<"relative tolerance: "<<relTol<<endl<<endl;

cout<<"\tf(a)\t\tf(p)\t\tf(b)"<<endl;

int N;

FMFLOAT minimum;

if (abs(a)<abs(b)) minimum=abs(a);

else minimum=abs(b);

N=log((b-a)/(absTol+minimum\*relTol))/log(2.0);

int i=1;

FMFLOAT p=0;

while (i<=N)

{

p=a+(b-a)/2;

if ((b-a)/2<=absTol+relTol\*abs(p)) break;

cout<<"round "<<i<<": "<<(\*f1)(a)<<"\t"<<(\*f1)(p)<<"\t\t"<<(\*f1)(b)<<endl;

if (sign(f1,a)\*sign(f1,p)>0) a=p;

else b=p;

i++;

}

cout<<endl;

cout<<"result for bisection method: "<<endl<<"total circulation number: "<<i<<endl<<"root found: "<<p<<endl<<"f(x): "<<(\*f1)(p)<<endl<<endl<<endl;

return p;

}

}

FMFLOAT rootfinding::Newton(FMFLOAT (\*f1)(FMFLOAT x), FMFLOAT (\*f2)(FMFLOAT y), FMFLOAT startValue, FMFLOAT absTol, FMFLOAT relTol)

{

cout<<"Newton function: "<<endl;

cout<<"starting point: "<<startValue<<endl;

cout<<"absolute tolerance: "<<absTol<<endl<<"relative tolerance: "<<relTol<<endl<<endl;

cout<<"\tf(p)\t\tfp(p)\tdel\t\tdel/p\t\tp"<<endl;

int i=1;

FMFLOAT p=startValue;

FMFLOAT del=-f1(p)/f2(p);

while ((abs(del)>(absTol+abs(p)\*relTol))&&(i<=10))

{

cout<<"round: "<<i<<": "<<(\*f1)(p)<<"\t"<<(\*f2)(p)<<"\t"<<del<<"\t"<<del/p<<"\t"<<p<<endl;

p=p+del;

del=-f1(p)/f2(p);

i++;

}

cout<<"round: "<<i<<": "<<(\*f1)(p)<<"\t"<<(\*f2)(p)<<"\t"<<del<<"\t"<<del/p<<"\t"<<p<<endl<<endl;

cout<<"result for Newton method: "<<endl<<"total circulation number: "<<i<<endl<<"root found: "<<p<<endl<<"f(x): "<<(\*f1)(p)<<endl<<endl<<endl;

return p;

}

FMFLOAT rootfinding::Solve(FMFLOAT (\*f1)(FMFLOAT), FMFLOAT (\*f2)(FMFLOAT), FMFLOAT a, FMFLOAT b, FMFLOAT absTol, FMFLOAT relTol)

{

FMFLOAT n;

n=this->Bisect(\*f1,a,b,absTol,5\*pow(10.0,-2));

return this->Newton(\*f1,\*f2,n,absTol,relTol);

}

<main.cpp>

#include "rootfinding.h"

#include "FMFloat.h"

#include <iostream>

#include <math.h>

#include <iomanip>

//#define FMFLOAT double

using namespace std;

FMFLOAT f(FMFLOAT i)

{

return pow(1+i,180)-1000\*i-1;

}

FMFLOAT fp(FMFLOAT i)

{

return 180\*pow(1+i,179)-1000;

}

void main()

{

rootfinding a;

FMFLOAT out=0;

FMFloat m;

FMFLOAT k=10;

out=a.Solve(\*f,\*fp,0.001f,0.1f,pow(10.0,-7),m.epsilon(out));

cout<<"the effective annual return rate is : "<<pow(out+1,12)-1<<endl<<endl<<endl;

cin>>out;

}