**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 20.692869% and the speed of the Newton method convergence is faster than Bisection and is nearly equal to 

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:

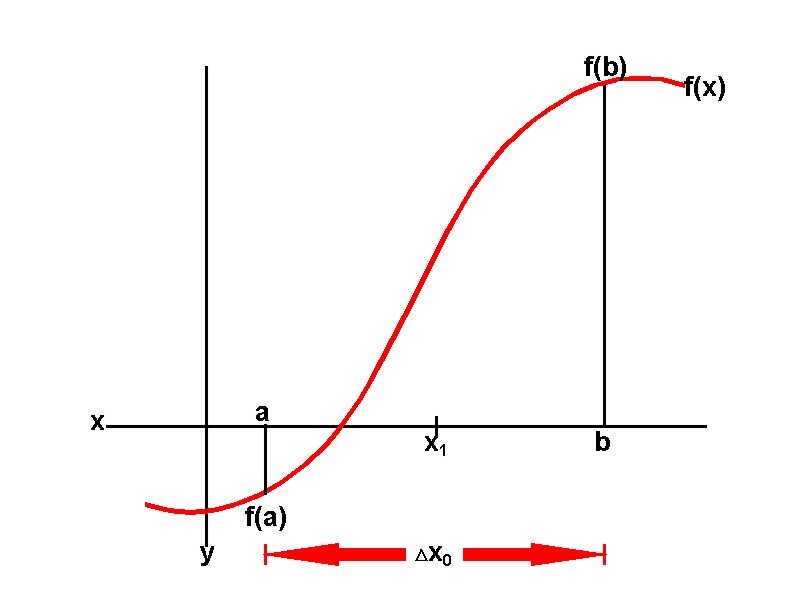
The bisection method requires two initial points a and b such that f(a) and f(b) have opposite signs.

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

Secondly, divided 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. Which is equal to the for the single perception

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.

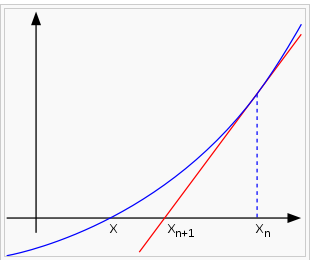


1. Newton’s method:

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 just like 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:



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:



1. **Results**

The Result of the Bisection Method under the one significant digit is as follows:

I choose the result of the Newton method as the true value, which is 0.0157967077420546 to derive the related errors.

The result of the Bisection Method is:

|  |  |  |
| --- | --- | --- |
| iteration Number | Int Rate | Rel Error |
| 1 | 0.5005000000000 | 3.07E+01 |
| 2 | 0.2507500000000 | 1.49E+01 |
| 3 | 0.1258750000000 | 6.97E+00 |
| 4 | 0.0634375000000 | 3.02E+00 |
| 5 | 0.0322187500000 | 1.04E+00 |
| 6 | 0.0166093750000 | 5.14E-02 |
| 7 | 0.0088046875000 | 4.43E-01 |
| 8 | 0.0127070312500 | 1.96E-01 |
| 9 | 0.0146582031250 | 7.21E-02 |
| 10 | 0.0156337890625 | 1.03E-02 |

Use the result of the Bisection method (0.0156337890625) as the initial value in the Newton method, the result can be seen from the following table:

|  |  |  |
| --- | --- | --- |
| The Newton Method(Single Perception) | |  |
| Iteration Number | Int Rate | Rel Error |
| 1 | 0.0157999202047 | 2.03E-04 |
| 2 | 0.0157966825922 | 1.59E-06 |
| 3 | 0.0157967079447 | 1.28E-08 |
| 4 | 0.0157967077404 | 1.03E-10 |

The Newton Method (Double Perception)

|  |  |  |
| --- | --- | --- |
| Iteration Number | Int Rate | Rel Error |
| 1 | 0.0157999202046632000 | 2.03E-04 |
| 2 | 0.0157966825922434000 | 1.59E-06 |
| 3 | 0.0157967079447340000 | 1.28E-08 |
| 4 | 0.0157967077404214000 | 1.03E-10 |
| 5 | 0.0157967077420679000 | 8.38E-13 |
| 6 | 0.0157967077420546000 | 2.86E-15 |

From the above table, we can see that the convergent speed of Newton method is faster than the Bisection Method. Just as I listed above in the description of the mathematics,

. **My results are in accordance with the prediction, which can be the proof that they are right.**

Finally, using the formula  to get effective annual return rate: 20.692869%

1. **Conclusions**

The final result of the annual interest rate is 20.692869% (pretty good return☺) and we can find the speed of the convergence of Newton Method is faster than the Bisection Method.

The speed of the related error of Newton method is 

1. **Code Listing**
2. The method of Bisection

Sub Bisection()

Dim p, n, reltol, abstol, f, fb As Double

Dim a As Double

Dim b As Double

Dim int\_rate As Double

n = 180

a = 0.001

b = 1

abstol = 10 ^ (-38)

reltol = 0.5 \* 10 ^ (-1)

Do

int\_rate = (b + a) / 2

fa = Amount(a)

fp = Amount(int\_rate)

If fa \* fp > 0 Then

a = int\_rate

Else

b = int\_rate

End If

i = i + 1

Worksheets("sheet2").Cells(i, "a").Value = int\_rate

Loop While (b - a) / 2 > abstol + reltol \* int\_rate

End Sub

1. The Method of Newton

Sub Newton()

Dim p, n, reltol, abstol, f, fb As Double

Dim a As Double

Dim b As Double

Dim int\_rate As Double

Dim condition As Double

int\_rate = 0.0156337890625

n = 180

a = 0.001

b = 1

abstol = 10 ^ (-38)

reltol = 0.5 \* 10 ^ (-7)

Do

fa = Amount(int\_rate)

fp = Amount\_derivative(int\_rate)

int\_rate = int\_rate - fa / fp

i = i + 1

Worksheets("sheet2").Cells(i, "b").Value = int\_rate

If fa / fp > 0 Then

condition = fa / fp

Else: condition = -fa / fp

End If

Loop While condition > abstol + reltol \* int\_rate

End Sub

1. The function of the Total Amount

Function Amount(x As Double) As Double

Amount = 1000 / x \* ((1 + x) ^ 180 - 1) - 1000000

End Function

1. The function of the Derivative of the total amount

Function Amount\_derivative(x As Double) As Double

Amount\_derivative = 1000 \* (180 \* ((1 + x) ^ 179) \* x - (1 + x) ^ 180 + 1) / (x ^ 2) - 1000000

End Function