Financial Calculator App – Analysing the Efficiency and Runtime of Multiple IRR Methods

Seán Ryan
B.A.(Mod.) Business and Computing
Final Year Project April 2013
Supervisor: Mary Sharp

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

______________________________________________        ________________________
Name            Date
Acknowledgements

I would like to sincerely thank my supervisor Mary Sharp for her help and support during this project.
# Table of Contents

Introduction .......................................................................................................................... 5

1.1 Motivation ........................................................................................................................ 5

1.2 Project Aims ...................................................................................................................... 6

1.3 Technical Approach ......................................................................................................... 6

1.4 Outline .............................................................................................................................. 7

Background .................................................................................................................................. 8

2.1 Introduction to Background ........................................................................................... 8

2.2 Financial Formulas .......................................................................................................... 8

2.2.1 Present Value .............................................................................................................. 9

2.2.2 Future Value ............................................................................................................... 10

2.2.3 Present Value of an Annuity ..................................................................................... 10

2.2.4 Future Value of an Annuity ..................................................................................... 12

2.2.5 Present Value of a Growing Annuity ....................................................................... 13

2.2.6 Present Value of a Perpetuity .................................................................................. 13

2.2.7 Present Value of a Growing Perpetuity ................................................................... 14

2.2.8 The Convexity of a Bond ....................................................................................... 14

2.2.9 Macaulay Duration ................................................................................................. 15

2.2.10 The Internal Rate of Return .................................................................................. 15

2.2.10.1 The Secant Method ......................................................................................... 16

2.2.10.2 The Newton-Raphson Method ........................................................................ 17

2.2.10.3 The Bisection Method ..................................................................................... 18

2.3 Literature Review .......................................................................................................... 18

2.4 State of the Art .............................................................................................................. 20
Table of Figures

Figure 2.1: Present Value Formula........................................................................................................9
Figure 2.2: Future Value Formula ........................................................................................................10
Figure 2.3: Present Value of an Annuity Formula ...............................................................................11
Figure 2.4: Example of Present Value of an Annuity ..........................................................................11
Figure 2.5: Future Value of an Annuity Formula ................................................................................12
Figure 2.6: Example of Future Value of an Annuity ..........................................................................12
Figure 2.7: Present Value of a Growing Annuity Formula ..................................................................13
Figure 2.8: Present Value of a Perpetuity Formula .............................................................................13
Figure 2.9: Present Value of a Growing Perpetuity Formula ...............................................................14
Figure 2.10: Convexity of a Bond Formula .........................................................................................14
Figure 2.11: Macaulay Duration Formula ..........................................................................................15
Figure 2.12: Internal Rate of Return Graph .........................................................................................16
Figure 2.13: Secant Method .................................................................................................................17
Figure 2.14: Newton-Raphson Method .............................................................................................18
Figure 2.15: Bisection Method ............................................................................................................18
Figure 3.1: Screenshot of Present Value Screen ................................................................................25
Figure 4.1: Screenshot of Convexity code ..........................................................................................27
Figure 4.2: Screenshot of Net Present Value code ............................................................................29
Figure 4.3: Screenshot of Differentiation Method ..............................................................................30
Figure 5.1: Iteration Results for average priced bond .......................................................................32
Figure 5.2: Runtime Results for average priced bond .........................................................................32
Figure 5.3: Iteration Results for expensive bond ................................................................................33
Figure 5.4: Runtime Results for expensive bond ................................................................................33
Figure 5.5: Results for Bisection Method of average priced bond .....................................................34
Chapter 1

Introduction

1.1 Motivation

There are many possible root-finding algorithms that are used to solve equations in mathematics. These types of methods are not calculated analytically, but iteratively in order to find the most accurate root of an equation. In relation to the world of finance, these root-finding methods are applied when trying to calculate the internal rate of return (IRR) of an investment. The internal rate of return can be imperative to investors when analysing whether or not to invest in a certain project. Therefore, analysing the accuracy of root-finding methods should be important to those who seek an accurate IRR of investments.

Determining not only the most accurate, but also the most efficient formula is also of importance. Examining the number of iterations a method uses until the desired accuracy has been reached is one potential way of examining the efficiency of a particular method. However, in modern times there are few investors, new or experienced, who are likely to determine the IRR without using some sort of technology. In that case, it is not enough to only examine the number of iterations of a formula. Instead, examining the runtimes associated with each method can determine the most efficient method of root-finding in relation to the IRR.
1.2 Project Aims

There are two main objectives in this research. The first objective is to create a more user-friendly financial calculator than the current products, such as those that are sold in shops and as applications online. In relation to the physical financial calculators that are sold in shops, the objective is to present the formulas in simpler terms such that those new to finance and investing can understand how to operate them. In terms of the financial calculators that are sold on mobile app markets, the aim is again to develop a calculator that is easier to use for new and experienced investors as well as deal with the presentation issues that occur in current financial calculator apps. The problems associated with current products are discussed in more detail in section 2.4 State of the Art.

The second objective of this project is to research the numerous ways of finding the IRR and analyse the accuracy and efficiency of each method. In order to do this, experimentations were conducted with several test cases against three root-finding methods. The experimentation is discussed in chapter five (Evaluation) of the report.

1.3 Technical Approach

The financial calculator app was programmed as an Android app, primarily using the language Java. This was mainly due to the ease at which it is possible to publish an app on the android market, market share that the android market possesses over its rivals, and the high standard of documentation available to android developers through a number of different websites such as the Google developers site.
1.4 Outline

Chapter 2 of the report is focused on the background of the project. This will include a short review of the literature that is related to the mathematics of the root-finding methods, root-finding in relation to IRR and programming algorithms that have been used to calculate IRR. The chapter will also include a review of the current products, outlining their problems and how this project will attempt to solve these issues, and an explanation of the financial formulas that are used in the project, including IRR and the methods of root-finding.

Chapter 3 will detail the design and layout of the financial calculator app. This will include an explanation of the technologies used and the use case of the app.

Chapter 4 concerns the implementation of the formulas. This chapter will detail how the financial formulas were programmed in the app and highlight the important methods that were used.

Chapter 5 will explain the experimentation that was undertaken to find the most efficient method of root-finding in relation to IRR. This will include the experimentation criteria, the results of the experimentation, and the findings that relate to each method.

Chapter 6 will conclude with the report with a summary of the project. This chapter will also detail the possible changes to the process and the potential for further work in the field.
Chapter 2

Background

2.1 Introduction to Background

In this chapter, there will be an explanation of all the financial formulas that were used in developing the financial calculator. As part of this section, there will also be a review of the three root-finding methods that were tested by the later experiment. These methods are the Secant method, the Newton-Raphson method and the Bisection method. There will then be a short review of the literature relating to root-finding methods and the technical aspects of programming financial calculations. The chapter will conclude with a review of the current products that are available. The review of physical financial calculators will be focused on the BA II Plus financial calculator made by Texas Instruments and the review of online financial calculators will focus on the app developed by NRS Magic Ltd. which is available on the Google Play app store.

2.2 Financial Formulas

This section will individually explain each financial formula that was used in making the financial calculator app. The formulas that were used were Present Value, Future Value, Present Value of an Annuity, Future Value of an Annuity, Present Value of a Growing Annuity, Present Value of a Perpetuity, Present Value of a Growing Perpetuity, Convexity of a Bond, Duration of a Bond and the Internal Rate of Return.
2.2.1 Present Value

The present value of an investment is defined by finance website Investopedia as “the current worth of a future sum of money or stream of cash flows given a specific rate of return” [1]. The present value formula is also known as the discounted value or the present discounted value. The mathematical formula for calculating present value is given in the figure 2.1 [2]. In this diagram, “C” is the future sum of money, “i” is the interest rate for the period and “n” is the number of period between the current value and the future value of the sum of money.

\[ PV = \frac{C}{(1 + i)^n} \]

Figure 2.1 – Present Value Formula

To a non-investor or someone who is unfamiliar with finance, this formula may sound more complicated than it really is. The basic idea behind this formula is that receiving €500 today is worth more than receiving €500 in a years’ time. This is because you could invest the money received today and potentially receive an additional return on your investment in one year’s time that would make it more valuable. It is used by people to analyse and compare potential investments. For example, what investment decision would you make if you were offered €1,750 in two years’ time with an interest rate of 17.5% or €1,500 in the same time period with an interest of 8%? Using the present value formula (e.g. \( 1,750 / (1.1)^2 \)) we get a value of €1,267 for the first investment and €1,286 for the second investment. In this case the investor would choose the first investment as they have invested less money and received a higher sum in the future.
2.2.2 Future Value

The future value of an asset is almost the direct opposite of present value and a lot easier to understand. The future value is the value of an asset at a specific point in the future \[3\]. The future value formula is given in figure 2.2 \[4\], where “PV” stands for the present value of an asset, “r” is the interest rate and “t” is the number of periods.

\[
FV = PV \cdot (1 + i)^t
\]

Figure 2.2 – Future Value Formula

For example, if someone was to invest €2,000 for 4 years at a 12.5% interest rate, they would have to use the future value formula to calculate the value of their investment at the end of the four year period. In this case \((2000 \times (1.125)^4)\), the future value of the investment at the end of the four years would be €3,203.61. This formula is also used by investors to compare potential investments.

2.2.3 Present Value of an Annuity

Before describing what the present value of an annuity formula is, it would be beneficial to begin with a description of what an annuity actually is, as there are many different formulas in the app that relate to it. An annuity is a stream of fixed payments over a specific period of time \[5\]. A pension is a form of an annuity where people buy into the annuity and then, when they retire, they receive fixed payments over a specific period of time. The present value of an annuity is, like the previous present value formula, the “current value of a set of cash flows in the future, given a specific rate of return or discount rate” \[6\]. The formula is given in figure 2.3, where “C” is the value of each cash flow, “i” is the interest rate and “n” is the number of periods.
So if you attempting to calculate the present value of an annuity which paid €1,000 for each period, with an interest rate of 5% for 5 periods then you would have to apply the above formula. Investopedia have a diagram which shows the calculations of the above problem done on a year to year basis [7].

Alternatively to that, you could also apply the formula as you know that each payment is fixed. In the above example, the calculations look like this: $1,000 \times (1 - (1.05)^{-5}) / 0.05 = 4,329.48$.
2.2.4 Future Value of an Annuity

The future value of an annuity is the value of a group of payments at a specific time in the future \([8]\). The formula is given in figure 2.5 \([9]\) with “\(r\)” representing the interest rate and “\(n\)” representing the number of periods.

\[
FV_{\text{annuity}} = \frac{(1 + r)^n - 1}{r} \cdot (\text{payment amount})
\]

Figure 2.5 – Future Value of an Annuity Formula

For example, if you wanted to find the future value of an annuity after five periods that pays out $1,000 per period and it had an interest rate of 5% then you could apply the above formula. The result would be: $1000 \times [(1.05)^5 - 1 / 0.05] = 5525.64$. The figure 2.6 shows this computation in simpler terms as it is done on a year by year basis.

Figure 2.6 – Example of Future Value of an Annuity
2.2.5 Present Value of a Growing Annuity

The present value of a growing annuity is very similar to the present value of an annuity, with only difference being that the periodic payments grow at a proportionate rate. The mathematical formula for this is shown in figure 2.7.

\[
\frac{P}{r - g} \left[ 1 - \left( \frac{1 + g}{1 + r} \right)^n \right]
\]

- \( P = \text{First Payment} \)
- \( r = \text{rate per period} \)
- \( g = \text{growth rate} \)
- \( n = \text{number of periods} \)

Figure 2.7 – Present Value of a Growing Annuity Formula

2.2.6 Present Value of a Perpetuity

A perpetuity is an annuity that continues indefinitely [10]. This means that it is a stream of payments which begin on a fixed date and continue without end. The formula for this is very simple and is given in figure 2.8 where “\( A \)” is the amount paid each period and “\( r \)” is the interest rate.

\[
PV = \frac{A}{r}
\]

Figure 2.8 – Present Value of a Perpetuity Formula
2.2.7 Present Value of a Growing Perpetuity

The present value of a growing perpetuity is the current value of a stream of payments that grow at a proportionate rate that continue indefinitely. The formula for this is given in figure 2.9.

\[ PV \text{ of Growing Perpetuity} = \frac{D_1}{r - g} \]

\[ D = \text{Dividend or Coupon at period 1} \]
\[ r = \text{discount rate} \]
\[ g = \text{growth rate} \]

Figure 2.9 – Present Value of a Growing Perpetuity Formula

2.2.8 The Convexity of a Bond

The convexity of a bond is a measure of the sensitivity of the duration of the bond to changes in interest rates [11]. This formula is used as a risk-management tool as it helps to measure the amount of market risk to which a portfolio of bonds is exposed to. The mathematical formula for convexity is given in figure 2.10 [12], where “\(B\)” represents bond price, “\(d\)” represents duration and “\(r\)” represents the interest rate.

\[ C = \frac{1}{B} \left( \frac{d^2(B(r))}{dr^2} \right) \]

Figure 2.10 – Convexity of a Bond Formula
2.2.9 Macaulay Duration

The Macaulay duration is a method for finding the weighted average term to maturity of the cash flows from a bond [13]. In simplified terms, it is the calculation of the average economic lifetime of bond. The Macaulay duration is calculated as follows.

\[
\text{Macaulay Duration} = \frac{\sum_{t=1}^{n} t \times C}{\text{Current Bond Price}} + \frac{n \times M}{(1 + y)^n}
\]

Where:
- \( t = \text{respective time period} \)
- \( C = \text{periodic coupon payment} \)
- \( y = \text{periodic yield} \)
- \( n = \text{total number of periods} \)
- \( M = \text{maturity value} \)

Figure 2.11 – Macaulay Duration Formula

2.2.10 The Internal Rate of Return

To know what IRR is, one must first understand Net Present Value (NPV). NPV, also known as Net Present Worth, is the sum of the present values of future cash flows that come from the same entity [14]. It is the difference of future cash inflows and outflows of a project. If the NPV is positive then the project should be invested in as it will provide a positive cash flow in the future. Conversely, if the NPV is negative then the project should be avoided. The NPV is therefore an effective method for comparing and contrasting potential investments.

The internal rate of return is also used to measure worth of potential investments. It is the discount rate at the point where the NPV of all cash flows equal to zero [15]. Generally, the
IRR is the discount rate at which the present value of the costs of a project is equal to the present value of the benefits of a project.

![Internal Rate of Return Graph]

*Figure 2.12 – Internal Rate of Return Graph*

Investopedia describes IRR as such – “You can think of the IRR as the rate of growth a project is expected to generate. While the actual rate of return that a project ends up generating will often differ from its estimated IRR rate, a project with a substantially higher IRR value than other available options would still provide a much better chance of strong growth.” [16]

There are many different formulas to calculate the IRR as it is an iterative root finding method. Three methods to calculate the IRR are the Secant method, the Newton-Raphson method and the Bisection method.

### 2.2.10.1 The Secant Method
The secant method requires two initial estimates of the IRR to work. The formula is shown in figure 2.13 [17]. In this figure, "\(X_{n-1}\)" represents one guess of the IRR, "\(X_{n-2}\)" represents the other guess and the function of these figures is the NPV formula.

\[
x_n = x_{n-1} - \frac{f(x_{n-1})}{f(x_{n-1}) - f(x_{n-2})}
\]

Figure 2.13 – Secant Method

This formula is repeated with the result taking the place of one of the initial roots. E.g. in the example above, after \(X_n\) has been computed, \(X_{n-2} = X_{n-1}, X_{n-1} = X_n\). So the first initial guess is discarded, with the result taking its place. This process continues until the NPV of an IRR is equal to zero or the difference between the IRR found and the previous IRR is within a certain pre-set range (i.e. if looking for an accuracy of 0.00001, then subtract the IRRs and stop iterations when the difference is less than the above number.)

2.2.10.2 The Newton-Raphson Method

The Newton-Raphson method only requires one initial estimate to find the IRR. However, the Newton-Raphson method also requires the first derivative of the function, in this case the IRR. That means that to perform the Newton-Raphson method, you must first differentiate the NPV formula. The formula is given in figure 2.14 [18] where "\(X_0\)" represents the initial approximation of the IRR, the function is the NPV formula, and the first derivative of the formula is the differentiated NPV formula. Once again, this method is repeated until the NPV of an IRR is equal to zero or a specified accuracy has been met.
2.2.10.3 The Bisection Method

The bisection method requires two initial guesses of the IRR. The method bisects the interval of the two guess, sets a new subinterval with its results and then continues to bisect the subintervals until NPV = 0 or a specific accuracy is achieved [19]. The formula is given in figure 2.15 in pseudo code form.

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \]

Figure 2.14 – Newton-Raphson Method

While [NPV (X) != 0 or Accuracy is achieved]

\[ A = \text{(negative) guess 1, } B = \text{guess 2} \]
\[ X = (A + B)/2 \]

If NPV(A) * NPV(X) < 0, then B = X
If NPV(A) * NPV(X) > 0, then A = X

Figure 2.15 – Bisection Method

2.3 Literature Review

This section is a short examination of the relevant literature concerned with root-finding algorithms and the best practices to use when attempting to program these methods.
The third chapter of Miranda and Facker’s “Applied Computational Economics and Finance”, deals with root-finding algorithms and explains the best practices for developing the methods in MATLAB [20]. The chapter outlines the disadvantages of using Newton’s method. It claims that the method can be very sensitive to the initial guess of the root, especially when the value of the initial estimate differs in significant size from the actual root. This can cause the first derivative of the function to change sign frequently. The chapter also notes that the derivative of the function can suffer rounding error problems, causing inaccuracy. They describe the bisection method as being the most simple and robust method for finding the root of an equation. The book notes that the bisection method is best used in conjunction with other root-finding methods. This is because the bisection method can take a large number of iterations to find the IRR, yet it is guaranteed to converge towards the root.

In the ninth chapter of “Numerical Recipes in C: The Art of Scientific Computing”, the authors explain root finding methods, outline their benefits and drawbacks and provide guidance for programming the methods in C [21]. The authors agree with Miranda and Facker’s assessment of the bisection method in that it will always converge towards the root, however, it will do so in a long period of time. They note that “Unfortunately, the methods that are guaranteed to converge plod along most slowly, while those that rush to a solution in the best cases can also dash rapidly to infinity without warning”. The secant method is identified by the authors as one method which is very quick yet is not always guaranteed to converge. The authors also agree with Miranda and Facker’s description of the Newton-Raphson method as being very sensitive to the initial estimate and the effects it can have on the first derivative of a function. The authors are also of the opinion that the secant method is a better root-finding finding method than the Newton-Raphson, because the Newton-Raphson requires you to find the first derivative of a function – “Therefore, the Newton-Raphson with numerical derivatives is always dominated by the secant method”.

The book “Java Methods for Financial Engineering: Applications in Finance and Investing”, is an overall guide as to how to program financial formulas [22]. The book details the root-
finding algorithms that are used to find the IRR. The book recommends using Java packages such as CoreMath, BaseStats and FinApps for programming root-finding algorithms. These packages have in-built functions that allow the programmer to avoid writing their own methods for some mathematical formulas. E.g. “YieldBisect” is an in-built method that can compute the bisection method given the correct parameters.

A short business paper written by Corality, a multinational financial consulting firm, details how to use the Newton-Raphson method in regards to finding the IRR [23]. The paper writes mathematical formulas and functions in terms of IRR, E.g. instead of describing IRR as being correct when \( f(x) = 0 \), like many other papers, this author notes that IRR is correct when \( NPV(IRR) = 0 \). This paper recommends initial estimates for each root-finding algorithm. The estimates were 10% for Newton-Raphson, 9.5% and 10.5% for the secant and 10% and 50% for the bisection method.

2.4 State of the Art

The current financial applications available today come in two varieties. These are (1) physical financial calculators that are sold in shops and (2) online financial calculators and financial calculator apps. Both of these have significant advantages but also great disadvantages that would deter users, especially those new to finance and investing.

2.4.1 Physical Financial Calculators

The main advantage associated with the financial calculators that are sold in shops is that they can perform a large amount of calculations, more than any financial calculator app. However, due to the depth of these functions, it can be very difficult to understand how to use the calculator for some formulas. E.g. the BA Plus II financial calculator has a manual
over 100 pages long with the vast majority of it dedicated to how to use financial formulas on the calculator [24]. It can confuse the user as to where they should enter certain figures. E.g. on the previously mentioned BA Plus II calculator, after pressing the button for IRR calculations, the screen reads “IRR =”, and then expects the user to input some data. However, it is difficult to know what data the user should input as there are no marked fields. This can be very confusing for all types of users, not just those new to finance. On online financial calculators and apps, the fields a user should input are clearly defined and laid out. The other main disadvantage is the cost of these calculators, as many online financial calculators can perform the same calculations for free of charge.

2.4.2 Online Financial Calculators / Financial Calculator Apps

Online financial calculators, such as the ones on websites like Investopedia [25] and financeformulas.net [26], can also perform a large quantity of formulas. However, these are likely to be a collection of calculators over a number of webpages as opposed to one calculator that can perform a variety of different calculations. It would also be a little more complicated for users to do these complications on a mobile device, as there are so many different calculators to choose from these sites.

Financial apps that are sold on the Apple app store and the Google Play Android app store do not contain as many financial formulas as either the physical financial calculators or the online financial calculators. Nevertheless, it does have the advantage of being one calculator, as opposed to many, and being only one click away on a mobile device or tablet. The user interface of these financial calculator apps, however, makes them difficult to use as they do not scale particularly well. E.g. the financial calculator developed by NRS Magic Ltd. seems to be developed solely for top of the range smartphones [27]. NRS appear to have forgotten those that have an interest in finance and do not own top smartphones, usually business or economics students. When using the app on a budget smartphone or a smartphone with a screen of smaller size than the top of the range smartphones, text
becomes squashed on the screen and words appear over two to three lines. They are also usually targeted at those who already have an understanding of finance, and not those that are new to the concept. E.g. financial calculator apps often use acronyms for the names of financial formulas.
Chapter 3

Design

3.1 Introduction to Design

This chapter will discuss the requirements for the projects, detailing what was needed to develop and test the android app. The chapter will then conclude with an overview of the layout of the app as well as an explanation of the use case.

3.2 Requirements

There were only basic requirements needed to develop this application. As the app was programmed in Java, Eclipse Juno was used as the development environment as it came with built-in Android Developer Tools and Android Software development kit (SDK).

To test the app both an android emulator and an android phone were needed. Android emulators are provided by the android SDK, where the app could be tested against a variety of different android versions and screen sizes. This was important to check the app for scalability issues, as the other financial apps mentioned in the state of the art section suffer from these problems. To make sure the app would work on an android phone, a Samsung Galaxy Europa running Android 2.2 was primarily used to test the application.
Using a java package like FinApps, which was discussed in the Literature review section, was considered. However, FinApps proved to be an unnecessary package as all the financial formulas that needed to be programmed could all be done using the basic in-built java maths functions. It was decided that using a package like FinApps could potentially slow down the runtimes of the root-finding methods. Therefore, it was decided that the methods should just be programmed manually.

3.3 Layout and Use Case

As one of the main objectives of this project was to make the app very user friendly and eliminate the problems that occur in other financial calculator apps, the financial calculator app was given a very basic design layout. This was to ensure that users who are not as finance-literate will be able to understand the app and its formulas. Therefore, the app was designed with a simple list view where all the financial formulas, written mostly in longhand (as opposed to financial calculator apps listed in the state of the art), were available to choose on the menu screen. When a user selects the formula he/she wants to calculate, they are shown a new screen with the heading of the formula, a large grey non-clickable and non-editable box, the criteria for the formula listed alongside a user-editable box and a button to calculate the formula.

To make it easier for the user and to help with the scalability issues, a number of different linear layouts were used in XML to make sure that the screen did not look cramped with edit boxes and buttons. The main issue identified with current financial calculator apps were that they did not scale well due to many edit boxes, headers, drop-down menus and buttons being placed on one line. Therefore, having a number of different linear layouts allows for there to be more control over what items get placed alongside each other. In this case, the only necessity was that a header and an edit box for the user were placed on the same line. E.g. if a method needed to know an interest rate to calculate a formula, both the header “Interest Rate” and an edit box for the user would be placed on the same line. Every other
text field or button was displayed on one line each. Once all the fields have been entered, the user presses the “Calculate” button which calculates the formula and prints the answer in the large grey box at the top of the screen. A screenshot of the present value formula page is given in figure 3.1.
Chapter 4

Implementation

4.1 Introduction to Implementation

This chapter will discuss how the financial formulas were programmed. After this there will be an explanation of the initial approach to programming the IRR. This was a method that was initially used in order to further understand IRR. It was not an algorithm that was used in the final financial calculator app. The chapter will conclude with a description of how the root-finding algorithms were programmed in relation to IRR.

4.2 Implementation of Financial Formulas

The implementation of the majority of the financial formulas was simply done by following the mathematics of the methods. E.G. for the present value formula, the user enters the future value of the sum of money, the interest rate and the number of periods. When all these figures are collected from the user, the formula for present value (figure 2.1 in the background to financial formulas section) can be implemented using the following line of code:

\[
PV = FV \times \left(\frac{1}{\text{Math.pow}(1+\text{rate}, \text{period})}\right);
\]

Formulas like the convexity of a bond and the Macaulay duration were more difficult to implement. This is because both methods require a number of iterations and hence, implementing a formula in one line, like the example above, is not an option. Both convexity and Macaulay duration are also not just about finding one particular figure. Before finding
the correct answer for the Macaulay duration, the present value of the bond must first be calculated before the formula can be applied. In the case of convexity, the Macaulay duration must first be calculated before the convexity can be calculated. E.g. Figure 4.1 shows the code for finding the convexity, where the Macaulay duration (double a) is found first, before the convexity formula (result) is used.

```java
87    per2 = period * -1;
88    PVAnuity = Price * (((1 - (Math.pow(1 + rate, per2))) / rate));
89
90    //py of parvalue
91    PVal = 1000 * (1 / (Math.pow(1 + rate, period)));
92    PVbond = PVal + PVAnuity;
93
94    x = ((Price + 1000) * period) / (Math.pow(rate + 1, period));
95    result = result + x;
96    period--;
97
98    while (period != 0)
99    {
100       x = (Price * period) / (Math.pow(rate + 1, period));
101          result = result + x;
102          period--;
103    }
104    double a = result / PVbond;
105
106    result = (1 / Price) * (((Math.pow(a, 2)) * (Price * rate)) / (a * Math.pow(rate, 2)));
```

Figure 4.1 – Screenshot of Convexity code

The approach to finding formulas such as convexity and Macaulay duration is similar to the other financial formulas, like the present value formula, in the way that it follows the correct financial mathematics. Nevertheless, it is more difficult to implement formulas such as convexity as there are a number of calculations that must first be solved in order to compute the desired answer.
4.3 Initial Approach to Internal Rate of Return

The initial approach to finding the internal rate of return was not to use any of the root-finding methods but to keep changing the value of the IRR until NPV was equal to zero. This approach used the following steps.

- **Step 1** – Make an approximation of the IRR and insert it into the NPV formula.
- **Step 2** – If the NPV is less than zero then decrease your approximation by 1, 0.1, 0.01, 0.001 or 0.0001 depending on the size of the NPV of your approximation. If the NPV is larger than zero then add to your approximation by the same values above.
- **Step 3** – Use the new approximation for IRR in the NPV formula
- **Step 4** – Continue the process until NPV is equal to zero

This method was not used in the final app, nor was it ever considered to be an accurate or efficient method for root-finding. This method had a very long runtime and often was inaccurate in its results. It is, however, a good starting point into understanding and developing root-finding methods for the IRR.

4.4 Implementation of Internal Rate of Return

It was decided that the simplest way to develop the IRR was to program the method for vanilla bonds. Vanilla bonds are one of the most common types of bonds that investors buy. They have a fixed return and pay back €1000 at maturity on top of the money made during the lifetime of the bond. The reason for this decision was that it is difficult to predict the potential return on stocks as the rate changes so frequently. Therefore, new investors are likely to only want to find the IRR of bonds they may be considering investing in, not stocks or other investments.
To implement the IRR, first the NPV method had to be programmed. The NPV formula is used within all the root-finding algorithms. As the IRR was developed for vanilla bonds, the coupon value (the amount paid per period) would always be fixed, apart from one payment that would be €1,000 greater because of the added return at the maturity of the bond. The code for the NPV formula is shown in figure 4.2.

```java
public static double npv(double pay, double per, double rat, double buyPrice) {
    rat = rat / 100;
    double x = 0;
    x += (1000 + pay) / (Math.pow(1 + rat, per));
    per--;
    while (per != 0) {
        x += (pay) / (Math.pow(1 + rat, per));
        per--;
    }
    x = x - buyPrice;
    return x;
}
```

**Figure 4.2 – Screenshot of Net Present Value code**

For each root-finding algorithm an initial estimate of the IRR must be taken. According to the Corality paper in the Literature Review section, the estimates for each algorithm are as follows: 9.5% and 10.5% for the secant method, 10% for the Newton-Raphson method and 10% and 50% for the bisection method. These initial estimates were implemented in each method due to the success in the Corality paper. Each method also implemented a check for the difference between the current approximation of the IRR and the previous estimate in order to achieve the desired accuracy. The accuracy chosen for these methods was no more than a difference of 0.000001 between the two estimates. This accuracy was chosen as it allows methods to find a close approximation of the IRR without going through a large amount of iterations for simple calculations.
Through use of the NPV formula, each root-finding method could simply be programmed by following its mathematical formula. However, the Newton-Raphson method required an additional method. To compute the Newton-Raphson formula you must also differentiate the NPV formula. IRR approximations must then be applied to the first derivative of the NPV formula. The code for this differentiation is shown in figure 4.3.

```java
public static double DevNPV(double pay, double per, double ret, double buyPrice)
{
  ret = ret / 100;
  double total = 0;
  double temp = per;
  double result = 0;

  for(int i = 0; i <= temp; i++)
  {
    if (i != 0)
    {
      if (i == temp)
      {
        result = ((pay=1000) " (i+1)) " Math.pow((1 + ret), (-i - 1));
      }
      else
      {
        result = (pay " (1 + i)) " Math.pow((1 + ret), (-i - 1));
      }
    }
    else
    {
      result = 0;
    }

    total = total - result;
    per--; } 

  return total; }
```  

Figure 4.3 – Screenshot of Differentiation Method

In mathematical terms, the differentiation of the NPV formula is as follows:

\[
-Coupon(1 + rate)^{-period-1} \ldots - [2 \times Coupon(1 + rate)^{-period-2}] \ldots - [n \times Coupon(1 + rate)^{-period-n}]
\]
Chapter 5

Evaluation

5.1 Introduction to Evaluation

This chapter concerns the experimentation conducted into finding the root-finding method with the quickest runtime and least amount of iterations. The test cases for the experiment will be explained as will the results. This chapter will conclude with an evaluation of the three root-finding methods in regards to the results found.

5.2 Experimentation Explained

The purpose of the experiment was to determine the most efficient method of IRR. For each method, twelve test cases were run a total of ten times each. The algorithms were run against an average priced bond of €1,000 and an expensive bond of €2,000. Each bond paid put semi-annually. They were also tested as a short-term bond of three years (six periods), a medium-term bond of eight years (sixteen periods) and a long-term bond of 20 years (40 periods). The coupon rate tested was a low coupon rate of 1% (€5 per period of the average priced bond, €10 per period of the expensive bond) and a high coupon rate of 14% (€70 per period of the average priced bond, €140 per period of the expensive bond). To find the number of iterations for each method, these test cases were only run once. To find the average runtime, the cases were run ten times each, in case of any discrepancies.
5.3 Results

The first test was to find the number of iterations for the average priced bond of €1,000. The resulting table for the secant and Newton-Raphson methods is shown in figure 5.1. In said table, the coupon received per period and number of periods is represented in brackets. E.g. (5,6) means that the bond paid out €5 per period over six periods.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Coupon, Periods)</td>
</tr>
<tr>
<td>Secant</td>
</tr>
<tr>
<td>Newton-Raphson</td>
</tr>
</tbody>
</table>

As we can see from the above table, the secant method uses a lower amount of iterations on average than the Newton-Raphson. However, the results also show that in some of the test cases, both methods have the same number of iterations. Hence, evaluating the runtimes of the two methods is a more accurate reflection of efficiency. The table with the resulting runtimes is given in figure 5.2.

<table>
<thead>
<tr>
<th>Time (in Milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Coupon, Periods)</td>
</tr>
<tr>
<td>Secant</td>
</tr>
<tr>
<td>Newton-Raphson</td>
</tr>
</tbody>
</table>
The results show that, even when the secant and Newton-Raphson methods are equal in the number of iterations, the secant method calculates the IRR at a faster speed. The results for the expensive bond of €2,000 produce similar results.

<table>
<thead>
<tr>
<th>(Coupon, Period)</th>
<th>(10,6)</th>
<th>(10,16)</th>
<th>(10,40)</th>
<th>(140,6)</th>
<th>(140,16)</th>
<th>(140,40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secant</td>
<td>11</td>
<td>42</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td>13</td>
<td>15</td>
<td>123</td>
<td>11</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 5.3 – Iteration Results for expensive bond

<table>
<thead>
<tr>
<th>(Coupon, Period)</th>
<th>(10,6)</th>
<th>(10,16)</th>
<th>(10,40)</th>
<th>(140,6)</th>
<th>(140,16)</th>
<th>(140,40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secant</td>
<td>1.0 – 1.4</td>
<td>1.7 – 2.6</td>
<td>1.1 – 2.0</td>
<td>0.9 – 1.4</td>
<td>1.1 – 1.3</td>
<td>1.3 – 1.5</td>
</tr>
<tr>
<td>Newton-Raphson</td>
<td>2.8 – 3.0</td>
<td>4.4 – 6.8</td>
<td>NA</td>
<td>2.7 – 3.4</td>
<td>3.5 – 3.6</td>
<td>5.2 – 5.7</td>
</tr>
</tbody>
</table>

Figure 5.4 – Runtime Results for expensive bond

One of the most telling results was that for a coupon payment of €10 per period over sixteen periods, the secant method used 42 iterations to find the IRR. This was twenty-seven more iterations than the Newton-Raphson. Nevertheless, the secant method still computed the figure in less time. The Newton-Raphson method struggled to compute the IRR when a coupon of €10 per period was paid over forty periods. This led to massive fluctuation in its runtime, with a wide range between the shortest and longest time it took to compute. Therefore, a record was not taken of the runtime as it was deemed irrelevant to the eventual findings emanating from the results of the experimentation.
5.4 Findings

5.4.1 Findings Related to the Bisection Method

The results from the bisection method were not included in the results section. This is because the runtime and number of iterations of the bisection method could not be compared to the results of the other two methods. If the purpose of this project is to find the most efficient method of IRR then the results of the bisection method from the first set of test cases immediately ruled it out. The results of the bisection method for the average priced bond of €1,000 can be seen in figure 5.5.

<table>
<thead>
<tr>
<th>(Coupon, Period)</th>
<th>(5,6)</th>
<th>(5,16)</th>
<th>(5,40)</th>
<th>(70,6)</th>
<th>(70,16)</th>
<th>(70,40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Runtime (m)</td>
<td>4.8 – 12.2</td>
<td>5.0 – 8.2</td>
<td>6.8 – 9.5</td>
<td>4.8 – 7.2</td>
<td>5.2 – 8.1</td>
<td>5.1 – 7.5</td>
</tr>
</tbody>
</table>

Figure 5.5 – Results for Bisection method of average priced bond

There is a significant difference between the shortest runtime of the bisection method and the largest, in one case the difference is 7.4 milliseconds. The longer runtime of the bisection is mainly due to the large amount of iterations it takes to find the IRR. Each test took 26 iterations, a greater amount than the average number of iterations it took the Newton-Raphson and secant methods.
5.4.2 Findings Related to the Newton-Raphson Method

The Newton-Raphson method produced slightly more iterations on average than the secant method, but greater runtimes in all test cases. The larger number of iterations is in direct conflict with the research behind the use of Newton-Raphson for finding IRR. The Corality business paper, detailed in the literature review section, came to the conclusion that the Newton-Raphson method only needed four iterations to find the IRR. However, the paper also stated that “Since we are unable to theoretically calculate the first order derivative of the NPV function, an estimate of the slope can then be found by taking the slope of two sufficiently close points on the NPV curve.” In the case of bonds, the NPV can be calculated. The Corality paper did not calculate the first derivative of the NPV formula and was instead estimating its value. Although this could potentially lead to less number of iterations, and a shorter runtime, the method would sacrifice accuracy for efficiency. If the first derivative of the NPV is being estimated, then the value will not be as accurate as the value of the derivation had the NPV been differentiated. To find a more accurate IRR using the Newton-Raphson method, the first derivative of the NPV formula must be calculated. Alternatively, to find the IRR in less iteration, the Newton-Raphson method where the first derivative of the NPV is estimated should be used.

It is this method, the differentiation of the NPV formula that causes the runtime to increase. The method for finding this differentiation is only used in the Newton-Raphson method. This is a disadvantage in relation to the secant method. In mathematics, the Newton-Raphson method is the preferred formula due to common belief that a root will be found in less iterations than other methods. However, in regards to the NPV, the number of iterations is higher using the Newton-Raphson method. For this reason, it is more beneficial to use the secant method for calculating IRR.
5.4.3 Findings Related to the Secant Method

The secant method found the IRR with the fastest runtime in all test cases and with less iterations than the other methods in the majority of the test cases. One of the main factors prominent in the secant method’s success was that it only needed to compute one method, the NPV, in order to find the IRR. The secant method only struggled to find the IRR of the expensive, medium-term bond with a low coupon rate. Despite the large number of iterations needed (42), the runtime was still quicker than the other two methods. The speed at which the secant method can calculate the IRR means that it is the most efficient root-finding method for financial calculators.
Chapter 6

Conclusion

6.1 Summary

The two main objectives outlined in the introduction were to develop a financial calculator app that was more user-friendly for new investors and to research which root-finding algorithm has the quickest runtime. In regards to the first objective, an app was developed that did not suffer from the scaling issues encountered with other financial calculator apps outlined in the state of the art section of the report.

6.2 Future Work

There are many more root-finding algorithms that could be tested in an experiment for efficiency. Brent’s method and the Inverse Quadratic method are two other root-finding methods that could be used to find the IRR. The Newton-Raphson method could also be re-tested without finding the first derivative of the NPV and instead estimating its value, like the Corality paper suggested.

The financial calculator could also be expanded to include more financial formulas. There are a very large number of formulas on a number of different calculators on financial websites, such as Investopedia and those described in the state of the art section. If the financial calculator app was to expand, then the main objective would be to combine these formulas into one financial calculator mobile app.
References


