VU University Amsterdam

Master's Thesis Business Analytics

Hedging Embedded Options in Life Insurance Products

Author:
Jonathan Moll

Supervisors:
Dr. R. Bekker (VU University)
Drs. P. van Zwol (SNS REAAL)
Ir. MSc. J. Minnaard (SNS REAAL)

March 2014

Preface

This thesis is written in partial fullfilment of the Master's program Business Analytics at VU University Amsterdam. The student has to do an internship and perform research on a practical problem of the internship granting company.

The internship granting company for this thesis is SNS REAAL, a Dutch bank-insurer with a number of brands. The internship was done at the Assets and Liabilities Management department of the Insurance branch of SNS REAAL (ALM Risk Insurance). Some brands of SNS REAAL have sold life insurance and pension products that contain options and guarantees. The value of these embedded options is exposed to market risks. The department ALM Risk Insurance is, among others, responsible for managing the risks connected to the embedded options in the life insurance and pension products that are sold.

First, I would like to thank my first supervisor at SNS REAAL, Pieter van Zwol, for his continuous support and for all time and effort he spend supervising me. I also would like to thank my second supervisor at SNS REAAL, Jeroen Minnaard. The friendly and useful conversations we had once a week were of great value to help me find my way in the company and kept me on track. I also want to thank my supervisors at VU University, René Bekker and Mark Hoogendoorn, for reading my thesis. In particular, I would like to thank my first supervisor at VU University, René Bekker, for his careful reading, the valuable feedback and the supportive conversations we had.

Jonathan Moll
March 2014

Abstract

Life insurance products often contain embedded options in the form of profit sharing and guaranteed returns on premiums deposited by the customer. The value of these embedded options is exposed to market risks. This thesis describes a method to find good hedge portfolios of swaptions to hedge the market risks of the embedded options. In addition, a method is described to monitor and clarify the hedge effectiveness based on the underlying risk drivers.

To find a good hedge portfolio of swaptions, an objective function is defined first. The objective value is based on four performance measures for the quality of a hedge portfolio. The performance measures quantify the mismatch between the swaptions and the embedded options in their sensitivity to different market risks. Two evolutionary algorithms (EAs) are used to compose hedge portfolios of swaptions that minimize the objective function. One EA searches for an optimal subset of swaptions within a given set of swaptions. This EA is used to unwind swaptions that are not effective anymore. The other EA searches for good swaptions to be added that will improve the portfolio. The results show that the algorithm generally produces good solutions with respect to the objective function and the four performance measures.

The hedge effectiveness is measured with the Dollar Offset method. The Dollar Offset method expresses the hedge effectiveness as the ratio of the change in value of the hedging instrument and the change in value of the hedged item. To clarify deviations from the desired effectiveness, we use a method to reconcile the changes of the swaptions and the embedded options, based on their underlying risk drivers. This gives insight in the causes of possible mismatches between the change in value of the swaptions and the change in value of the embedded options.

Contents

Chapter 1

Introduction

Different business units within SNS REAAL have sold pensions and life insurance contracts to their customers. These contracts contain embedded options and guarantees of which the value is subject to risks. One of the most influencing risks for the value of the embedded options is interest rate risk. This is the risk of losses arising from changes in interest rates and their volatilities. The actuarial half-yearly determines per business unit the value of the embedded options and their sensitivities to these changes.

The interest rate risk connected to the embedded options is hedged with swaptions, which is short for options on interest rate swaps. Swaptions are derivatives which are sold at financial markets. For each business unit, a hedge portfolio of swaptions must be constructed, based on the sensitivities of the embedded options. The swaptions that can be used for the hedge portfolio differ in strike price, time to expiry of the option, tenor of the underlying swap and type of the underlying swap (payer, receiver). Ideally, a combination of swaptions is chosen such that the sensitivities of the resulting portfolio is equal to the sensitivities of the embedded options. Moreover, the hedge portfolio should be robust in time, which means that the hedge portfolio should perform well over a considerable time horizon, without the need of frequent modifications to the portfolio. Finally, because swaptions cost money, a portfolio with less swaptions is preferable to a portfolio with the same sensitivities but with more swaptions.

In practice, the hedge portfolio of swaptions will often not replicate the sensitivities of the embedded options perfectly. Therefore, the hedge effectiveness of the portfolio should be monitored. This means that insight must be provided into the effectiveness of the hedge and where possible deviations from the expected effectiveness come from. For example, swaptions are more sensitive to the swap curve, while the embedded options are more sensitive to the government curve. Other risk drivers are changes in interest rates, changes in volatilities and the passing of time.

The problem posed above leads to two research questions.

1. How can a small hedge portfolio of swaptions be found, which replicates the sensitivities of the embedded options well and is robust in time?

2. How can the hedge effectiveness be measured and clarified, based on the underlying risk drivers?

SNS REAAL is constructing a tool called the ALM tool. The programming language used for the tool is PHP in combination with a MySQL database. The ALM tool is used for controlling market risks and has the functionality to import Excel data in the database, after which calculations can be executed and reports can be requested. The monitoring of the hedge effectiveness and the clarification of the hedge effectiveness based on underlying risk drivers must be implemented in the tool, such that a report of the hedge effectiveness can be requested on a frequent basis.

The remainder of this thesis is structured as follows. In Chapter 2, we pay attention to embedded options in life insurance products. We describe the products containing the embedded options and classify them based on their pay off structure. We also describe which risks are connected to embedded options and how SNS REAAL measures the sensitivities to these risks. In Chapter 3 we describe what swaptions are, how swaptions can be priced and to what extend they are sensitive to changes in interest rate risk. In Chapter 4, we develop an algorithm to find or improve hedge portfolios of swaptions to cover the sensitivities of the embedded options to interest rates and volatility. We first go into the objectives of the algorithm and describe the performance measures we defined for the quality of a hedge portfolio. We also define an objective function based on these performance measures. Then we design an algorithm that searches for good hedge portfolios of swaptions while keeping the number of swaptions used within some margin. In Chapter 5, we describe our implementation of the algorithm in Excel with VBA. We describe the structure of the application, go into the performance of the algorithm and present results. In Chapter 6 we discuss how the hedge effectiveness can be measured and how it can be clarified based on the underlying risk drivers. We also elucidate the report that we constructed in the ALM tool. Finally, in Chapter 7 we discuss our findings and draw conclusions.

Chapter 2

Embedded Options in Life Insurance Products

In this chapter we describe the products that are sold by the business units of SNS REAAL. These products contain embedded options that are subject to market risks. The aim of this chapter is to give the reader an idea of the products that are responsible for the risks that we are trying to hedge in this thesis. In Section 2.1, the different product types are elucidated. In the Section 2.2, we briefly describe how the value of the embedded options and guarantees of such products can be estimated. In Section 2.3, the method of SNS REAAL to monitor the sensitivities of the embedded options to market risk is described.

2.1 Products with Embedded Options

SNS REAAL sold different life insurance and pension products with embedded options. These products can be divided into three groups.

- Products with direct profit sharing.
- Products with cumulative profit sharing.
- Unit Linked products with guarantee.

In the next three subsections the different product groups are explained. The explanation is based on the descriptions of the Team Options and Guarantees [9].

2.1.1 Products with Direct Profit Sharing

2.1.2 Products with Cumulative Profit Sharing

2.1.3 Unit Linked Products with Guarantee

2.2 Valuation of Embedded Options and Guarantees

2.3 Risks

2.3.1 Sensitivities to Changes in Interest Rates

2.3.2 Sensitivities to Changes in the Volatility of Interest Rates

Chapter 3

Bonds, Swaps and Swaptions

The interest rate risk connected to embedded options can be hedged with swaptions, which are options on interest rate swaps. Swaptions are interest rate derivatives that are traded on financial markets. The purpose of this chapter is to explain what swaptions are, how swaptions are priced and to what extend they are sensitive to changes in interest rates. In Section 3.1, bonds are elucidated, because they form the basis for the theory about interest rate derivatives. In Section 3.2, swaps are described. In Section 3.3, swaptions are explained and a closed form formula for the value of a swaption is given. In Section 3.4, the risks connected to swaptions are discussed. The first three sections of this chapter are based on Björk [7].

3.1 Bonds

In this section we study bonds of various maturities. First, we look at zero coupon bonds, secondly, we show how zero coupon bonds can be used to price coupon paying bonds.

3.1.1 Zero Coupon Bonds

Definition 3.1. A **zero coupon bond** with maturity date T and face value FV, is a contract which pays the holder FV on date T. The price at time S of a bond with maturity date T and $FV = 1$ is denoted by $p(S, T)$.

The bond price $p(S, T)$ is a stochastic object with two variables, S and T, which are respectively the starting date and maturity date of the contract. For a fixed value of

S, $p(S,T)$ is a function of T. The graph of this function is called the zero curve or the term structure of money.

In contradiction to $p(S,T)$, the price $p(t,T)$ at time t, with $t \leq T$, is known and is given by

$$p(t,T) = \frac{1}{1 + r(t,T)(T-t)}, \tag{3.1}$$

where the zero rate $r(t,T)$ is the interest rate at time t for a bond with maturity T, which can be observed from the zero curve. The zero rate at time t expressed in terms of zero coupon bonds is given by

$$r(t,T) = \frac{1 - p(t,T)}{(T-t)p(t,T)}. \tag{3.2}$$

Suppose we invest in a contract at time t which allows us to buy a bond at time S that pays 1 at time T. We are interested in the interest rate that we get over the future interval $[S,T]$. The following experiment shows how such an interest rate can be found.

At time t, sell one bond with maturity S. This gives us $p(t,S)$. With this income we buy $p(t,S)/p(t,T)$ bonds with maturity T. At time S we are obliged to pay 1 and at time T we receive $p(t,S)/p(t,T)$. This means that a net investment of 1 at time S yielded $p(t,S)/p(t,T)$ at time T. Thus, at time t, the contract guarantees a riskless rate of interest over the future interval $[S,T]$. This interest rate is called a forward rate.
The forward rate $r_f(t,S,T)$ can be found by solving the following equation for $r_f(t,S,T)$:

$$1 + (T-S)r_f(t,S,T) = \frac{p(t,S)}{p(t,T)}. \tag{3.3}$$

The solution to this equation is

$$r_f(t,S,T) = \frac{p(t,S) - p(t,T)}{(T-S)p(t,T)}. \tag{3.4}$$

3.1.2 Coupon Bonds

Most bonds that are sold at the market with a longer maturity than 1 year are coupon bearing. Fortunately, coupon bonds can be priced in terms of zero coupon bonds, as we see in this subsection.

3.1.2.1 Fixed Coupon Bonds

The most common coupon bond is the fixed coupon bond. This bond pays a fixed amount to the holder of the bond at some intermediary points in time.

Definition 3.2. Denote a number of sequential points in time by $T_0, \ldots, T_n$. The holder of a **fixed coupon bond** with emission date T_0 receives deterministic coupons c_i at times $T_i, i = 1, \ldots, n$, and the face value of the bond FV at time T_n.

The price of a fixed coupon bond can be replicated by holding a portfolio of zero coupon bonds with maturities $T_i, i = 1, \ldots, n$. The replicating portfolio concists of c_i zero coupon bonds of maturity $T_i, i = 1, \ldots, n-1$ and $FV + c_i$ zero coupon bonds with maturity T_n. Thus, the price of the coupon bond $p(t)$, at time $t < T_1$, is given by

$$p(t) = FVp(t, T_n) + \sum_{i=1}^{n} c_i p(t, T_i). \tag{3.5}$$

In practice the coupons are often determined in terms of return, rather than in monetary terms. The return for the ith coupon is quoted as a rate acting on the face value FV, over the period $[T_{i-1}, T_i]$. Now, let $\alpha_i = T_i - T_{i-1}$ and let r_c be the rate of return. Then the price of the fixed coupon bond is given by

$$p(t) = FV \left(p(t, T_n) + r_c \sum_{i=1}^{n} \alpha_i p(t, T_i) \right). \tag{3.6}$$

3.1.2.2 Floating Rate Bonds

There are also coupon bonds for which the value of the coupon is not fixed at the time the bond is issued, but rather reset for every coupon period. The value of the reset coupon is most often determined by a financial benchmark, like a market interest rate.

Definition 3.3. Denote a number of sequential points in time by $T_0, \ldots, T_n$. The holder of a **Floating Rate Bond** with emission date T_0 receives the face value of the bond FV at time T_n, and variable coupons c_i at times $T_i, i = 1, \ldots, n$, that are determined by a financial benchmark.

Here, we discuss the floating rate bond where the coupon rate r_{c_i} is set to the zero rate. Let $\alpha_i = T_i - T_{i-1}$ again. Then the coupon rate r_{c_i} can be written as

$$r_{c_i} = \alpha_i r(T_{i-1}, T_i). \tag{3.7}$$

First, we want to express (3.7) in terms of zero coupon bonds. This can be achieved by plugging in Equation (3.2). We find that

$$r_{c_i} = \frac{\alpha_i(1 - p(T_{i-1}, T_i))}{\alpha_i p(T_{i-1}, T_i)} = \frac{1}{p(T_{i-1}, T_i)} - 1. \tag{3.8}$$

Secondly, we want to derive the value of (3.8) at time t. The value of a bond at time t, that pays you -1 at time T_i is $-p(t, T_i)$. So at time t, the second term on the right-hand side of (3.8) equals $-p(t, T_i)$.
The first term on the right hand side of (3.8), $\frac{1}{p(T_{i-1},T_i)}$, equals $p(t, T_{i-1})$ at time t. This can be shown with the following experiment.

Suppose at time t you buy a zero coupon bond with maturity T_{i-1}, so that your investment is $p(t, T_{i-1})$. At time T_{i-1} you will receive 1. You invest this amount in bonds with maturity T_i, which will give you $\frac{1}{p(T_{i-1},T_i)}$ bonds. At time T_i the bonds will mature, each at face value 1. Thus, at time T_i you will obtain $\frac{1}{p(T_{i-1},T_i)}$.

Given the values of the terms of Equation (3.8) at time t, we find that the value at t of r_{c_i} is $p(t, T_{i-1}) - p(t, T_i)$. Summing up the coupon rates and multiplying by the face value FV we find that the price at t of the floating rate bond is given by

$$p(t) = FV\left(p(t, T_n) + \sum_{i=1}^{n}[p(t, T_{i-1}) - p(t, T_i)]\right) = FVp(t, T_0). \tag{3.9}$$

3.2 Swaps

In the previous section two types of coupon bonds were described, the fixed coupon bond and the floating rate bond. An interest rate swap is a derivative where the payment streams of the fixed coupon paying bond are exchanged for the payment streams of the floating rate bond over the same face value. At predetermined sequential points in time $T_1, \ldots, T_n$ the coupons of the two payments streams are exchanged. The emission time of the contract is denoted with T_0. The fixed rate coupon, known as the swap rate, is denoted with r_{swap}. There are basically two types of swaps, payer swaps and receiver swaps. The terminology for swaps always refers to the fixed leg, so the holder of a payer swap pays the fixed leg and receives the floating leg. For the receiver swap it is the other way around.

The value of a swap is easily found by using (3.6) and (3.9) from the previous section. We find that the total value $\Pi(fixed, t)$ of the fixed leg at $t < T_0$ is given by

$$\Pi(fixed, t) = FV\left(p(t, T_n) - p(t, T_0) + r_{swap}\sum_{i=1}^{n}\alpha_i p(t, T_i)\right). \tag{3.10}$$

For the floating leg $\Pi(floating, t)$ this value is

$$\Pi(floating, t) = FV\left(p(t, T_0) - p(t, T_n) - r_{swap}\sum_{i=1}^{n}\alpha_i p(t, T_i)\right). \tag{3.11}$$

A fair swap rate at $t < T_0$ is chosen such that the value of the swap equals zero at the emission date of the contract. Thus, for $t < T_0$, a fair future swap rate r_f can be solved by setting (3.6) equal to (3.9)

$$FVp(t, T_0) = FV\left(p(t, T_n) + r_f\sum_{i=1}^{n}\alpha_i p(t, T_i)\right). \tag{3.12}$$

This leads to the following formula for the forward swap rate r_f.

$$r_f(t) = \frac{p(t, T_0) - p(t, T_n)}{\sum_{i=1}^{n}\alpha_i p(t, T_i)}. \tag{3.13}$$

The denominator of Equation (3.13) can be interpreted as the present value of the coupon payments if the coupons would pay 1 each. It is often referred to as the basis point value (bpv) of the swap,

$$bpv(t) = \sum_{i=1}^{n}\alpha_i p(t, T_i). \tag{3.14}$$

This basis point value is important, because with this value it is easy to compute the value of a forward swap contract with a predetermined fixed coupon rate. Suppose a contract gives you the possibility to enter into a future swap contract, which exchanges the floating zero rate r for a predetermined fixed coupon rate r_K. Assume that the current zero curve implies a forward swap rate $r_f(t)$. Then the current arbitrage free price for the future payer swap is

$$PS(t, r_K) = (r_f(t) - r_K)bpv(t)FV, \tag{3.15}$$

and the current price of the future receiver swap is

$$RS(t, r_K) = (r_K - r_f(t))bpv(t)FV. \tag{3.16}$$

The formulas above are also essential for the pricing of swaptions, which is the subject of the next section.

3.3 Swaptions

Swaption is short for option on an interest rate swap. A swaption is an interest rate derivative with the following definition.

Definition 3.4. A **swaption** gives you the right, but not the obligation to enter into a swap contract with a predetermined fixed coupon rate, with a predetermined **tenor**, on a predetermined date in the future, **the maturity date**.

The tenor of the swaption is the number of years the underlying swap contract lasts. Given Definition 3.4 and Equation (3.15) we can write the payoff $X(t)$ of a payer swaption as

$$\begin{aligned} X(t) &= max[PS(T_0, r_K); 0] \\ &= max[r_f(t) - r_K; 0] bpv(T_0) FV. \end{aligned} \tag{3.17}$$

The payer swaption is thus a call option on r_f with strike price r_K times a factor $bpv(T_0)FV$. This means that a swaption can be priced with a simple variant on the Black and Scholes formula known as Black's Formula for swaptions. Below, Black's formula for a payer swaption PSN and a receiver swaption RSN at time $t < T_0$ are given

$$\begin{aligned} PSN(t, r_K) &= \{r_f(t)N[d1] - r_K N[d2]\} \cdot bpv(t) \cdot FV \\ RSN(t, r_K) &= \{r_K N[-d2] - r_f(t, T_0, T_n)N[-d1]\} \cdot bpv(t, T_0, T_n) \cdot FV, \end{aligned} \tag{3.18}$$

where $N[\cdot]$ is the distribution function of the standard normal distribution and

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{(T_0 - t)}} \left[\ln\left(\frac{r_f(t)}{r_K}\right) + \frac{1}{2}\sigma^2(T_0 - t) \right] \\ d_2 &= d_1 - \sigma\sqrt{(T_0 - t)}. \end{aligned} \tag{3.19}$$

The constant σ is known as the Black volatility. If the market price for the swaption is given, this volatility can be derived from Black's formula. This volatility is called the implied Black volatility.

3.4 Risks

Chapter 4

Hedging Embedded Options With Swaptions

In Chapter 2, a way to valuate different types of embedded options in insurance products was described. The value of these embedded options is sensitive to changes in the shape of the yield curve. In Chapter 3 we discussed bonds, swaps and swaptions, which also draw their value from underlying interest rate curves. Since the value of interest rate products depend on interest rates, it makes sense to hold a portfolio of these products that approximately replicates the movements of the embedded options in the insurers liabilities. The purpose of such a portfolio is to hedge the interest rate risk of the embedded options. By holding a replicating portfolio of swaptions, the balance sheet can be stabilized. If the liabilities increase (decrease) due to interest rate changes, the swaptions on the assets side of the balance sheet will also increase (decrease).

In this chapter we describe how a small portfolio of swaptions can be found that replicates the sensitivities of the embedded options well and is robust in time. In Section 4.1 we explain what we exactly mean by a replicating portfolio and we specify the different objectives. In Section 4.2 we present an algorithm to find good solutions to the problem given the contradictory objectives. A perfect replication can never be found, but the solution will improve with the number of swaptions used. However, swaptions are costly and it is unwanted to hold a great number of swaptions. Thus, the solution is a trade off between the accuracy of replication and the transaction costs of the swaptions.

4.1 Objectives

4.1.1 Measuring Performance Based on KRD Sensitivities

4.1.2 Measuring Performance with Historic Interest Rate Changes

4.1.3 Measuring Performance Based on Volatility Sensitivities

4.1.4 Controling the Time Value Loss

4.1.5 The Objective Function

In the previous subsections we defined measures for the performance of a hedge portfolio of swaptions to cover the market risks of a portfolio of embedded options. The measures are chosen such that for all measures holds that the lower the value, the better the performance of the hedge portfolio. So we are searching for a portfolio of swaptions that minimizes a weighted sum of the performance measures described above. This leads to the following objective function that we want to minimize.

$$\text{Objective function: } w_{KRDE}KRDE + w_{HSE}HSE + w_{VE}VE + w_{TL}TL \tag{4.1}$$

The weights, w_{KRDE}, w_{HSE}, w_{VE} and w_{TL}, are numbers in $\mathbb{R}^+$. If the weights are normalised, the value of the objective function is just a weighted average over the four performance measures. How the weights are chosen depends on the situation and the preferences of the user(s).

The Second Objective

The objective function does not cover the preference of holding a portfolio with a small number of swaptions. This second objective is contrary to minimizing the objective function, because a portfolio with more swaptions is better capable to replicate the sensitivities of the embedded options. In order not to use too many swaptions, we add one swaption at a time and search for a solution. If one or more of the performances measures exceeds its threshold values, another swaption is added. This process is repeated until the criteria are met or the maximum number of swaptions is reached. In the next section, the algorithm is described in detail.

4.2 Algorithm Design

In Section 4.1, we defined an objective function for a hedge portfolio of swaptions, so that we can evaluate the performance of the portfolio and compare its performance with other portfolios. In this section we design an algorithm that searches for a good performing hedge portfolio, such that the number of swaptions in the portfolio does not become too large. We first give a description of the basic structure of the algorithm. The rest of the section is concerned with an important part of the algorithm that consists of minimizing the objective function by searching through a search space. We give a brief description of some well known search techniques and pay special attention to evolutionary algorithms. In Subsection 4.2.3 we design an evolutionary algorithm for the first part of the basic structure of our algorithm. Thereafter, in Subsection 4.2.4, we design an evolutionary algorithm for the second part of the basic structure of our algorithm.

4.2.1 Basic Structure

We want to optimize the performance of a hedge portfolio of swaptions to cover the market risks of a portfolio of embedded options. In practice, you start only once from scratch and thereafter you periodically decide whether the portfolio needs to be adjusted or not. This decision depends, of course, on the performance of the hedge portfolio. The performance of a hedge portfolio of swaptions can change, for example, because of a change in the embedded options portfolio or because of a swaption that almost matures and needs to be unwound. For the decision whether to adjust or not, we choose values for the four performance measures of Section 4.1 that may not be exceeded, threshold values. If one of the four performance measures exceeds its threshold value, the portfolio must be adjusted.

Once a swaption is bought, you cannot alter its properties. So the only way to adjust a portfolio is to unwind swaptions and/or to buy new ones. The logical first step is to try to make the portfolio better by unwinding swaptions. If one or more performance measures still exceed their threshold value(s), the next step is to improve the portfolio by adding a new swaption with well suited properties. Then, keep adding swaptions until the maximum number of swaptions is reached or all performance measures are below their thresholds.

In pseudo code, the basic scheme can be presented as in 4.1. The threshold values of the performance measures are denoted by T_KRDE, T_HSE, T_VE and T_TL. The number of swaptions in the portfolio is denoted with NUM_S and its threshold with MAX_NUM_S.

algorithm: `improvePortfolio`

if $KRDE > T\_KRDE$ **or** $HSE > T\_HSE$ **or** $VE > T\_VE$ **or** $TL > T\_TL$ **then**
    improveByUnwinding()
end if
while ($KRDE > T\_KRDE$ **or** $HSE > T\_HSE$ **or** $VE > T\_VE$ **or** $TL > T\_TL$) **and** ($NUM\_S < MAX\_NUM\_S$) **do**
    improveWithNewSwaption()
end while

FIGURE 4.1: Basic structure of the algorithm in pseudo code

As can be seen, we add one swaption at a time and search for a small value of the objective function by choosing well suited properties for only the last added swaption. An alternative is to add one swaption at a time and then look at all combinations of the properties of all the swaptions in the portfolio. This will lead to better solutions, but since the search space grows exponentially with each added swaption, it quickly requires too much computation capacity and/or time for practical purposes.

4.2.2 Searching the Search Space

The basic structure presented in the previous section contains two different problems for which we need to search through a search space for a solution that improves the objective function (4.1). The function improveByUnwinding searches within a set of swaptions, for a subset that leads to a better value of the objective function. The function improveWithNewSwaption searches for good objective function values, by varying the swaption properties of a newly added swaption. Search algorithms are iterative procedures, which

at each iteration, evaluate one or more candidate solutions and select in a clever way the candidate solutions to be evaluated in the next iteration.

Search algorithms generally need at least four components:

- A representation of a solution.
- An evaluation function.
- A routine that selects candidate solutions to be evaluated.
- A termination condition.

Representation

The first step in the design of a search algorithm is to define a representation of a candidate solution. A representation is a mapping of a candidate solution to the actual search space. For example, given an optimization problem where the possible solutions are sets of 3 integers, a given set of 3 integers could be a representation of a solution, but one could also decide to represent a solution by a set of binary codes. In this case, the representation of the solution [13 6 16] would become [01101 00110 10000]. Any kind of representation is possible as long as there is a one-to-one mapping between the representation of a candidate solution and the actual candidate solution.

Evaluation

The evaluation function, also known as the fitness function or the objective function, quantifies the quality of a certain candidate solution. This makes it possible to compare different candidate solutions and select the best. Because the evaluation function makes it possible to compare the quality of different candidate solutions, it implicitly represents the problem to be solved.

Termination Conditions

Most search algorithms are designed to converge either to a global or local optimum. When these optima are known, a natural termination condition would be to stop when such an optimum is found. However, these optima are generally unknown. Moreover, there is often no guarantee to find an optimum and the algorithm may never terminate. Therefore, we must define termination conditions that eventually stops the algorithm.

Eiben and Smith [8] name the following commonly used options for this purpose:

- The maximally allowed CPU time elapses.
- The total number of evaluations reaches a given limit.

- The evaluation score does not improve for a given period of time.

Almost all search algorithms need a condition from the list above., or a similar one that is guaranteed to eventually stop the algorithm.

Routines to Select New Candidate Solutions

An important and core part of a search algorithm is the way it selects new candidate solutions to be evaluated in the next iteration. Below, the routines of some well-known search algorithms to select or generate candidate solutions are given. We distinguish between three types of search algorithms: exact search algorithms, local search algorithms and evolutionary algorithms.

4.2.2.1 Exact Search Algorithms

Exact methods always find the global optimal solution. However, exact methods are often only applicable to specific problems or require too much time to find a solution. The most common exact search algorithm that is applicable to a wide range of problems is the brute force method. The description below comes from [2].

Brute force

The brute force method is a very general technique that simply generates all possible solutions and picks the best solution. The brute force method always finds the global optimal solution, but can only be applied to problems with a relatively small search space. Brute force searches can be speeded up by cleverly using the knowledge about the problem at hand to reduce the search space. However, in most practical situations the search space is far too large for simply evaluating all options.

4.2.2.2 Local Search Algorithms

Local search algorithms start with a given candidate solution and search in the "neighborhood" of that solution for a better solution. From there, it searches again in the neighborhood to a good solution. This process repeats until no better solution can be found in the neighborhood of the current solution or another termination condition is satisfied.

The advantages of local search algorithms are that they can find good solutions within a relatively small time period and that they are applicable to a wide range of problems.

The main disadvantage is that they not necessarily find the global optimum, it depends on the initial starting point of the search. If there are many local optima, it is likely that the system ends up in a local optimum that could be far away of the global optimum. Some search algorithms are extended with some stochastic methods that make it possible to escape from a local optimum and explore the search space in another neighborhood, were possibly a better optimum than the current can be found. Below, we briefly explain some common variants of local search algorithms. The description of Gradient Descent is based on [5] and the descriptions of Hill Climbing and Simulated Annealing are based on [3] and [1].

Gradient Descent

Gradient descent is a local search algorithm that takes steps in the direction of the (approximate) negative gradient of the evaluation function. If the algorithm takes steps in the direction of the positive gradient, the method is known as gradient ascent.

Gradient descent makes use of the fact that if the multivariate evaluation function $F(x)$ is defined and differentiable in the neighborhood of a certain point α, then $F(x)$ decreases fastest if one goes from α in the direction of the negative gradient of F at α, $-\nabla F(\alpha)$. Thus, if $b = \alpha - \gamma \nabla F(\alpha)$ with γ small enough, then $F(\alpha) \geq F(b)$.

The search routine starts with a guess x_0 and evaluates the sequence $x_0, x_1, x_2, \ldots$ such that $x_{n+1} = x_n - \gamma_n \nabla F(x_n)$, $n \geq 0$. The routine assures that $F(x_0) \geq F(x_1) \geq F(x_2) \ldots$. So hopefully the sequence $x_0, x_1, x_2 \ldots$ converges to a local or global minimum. With certain conditions on the evaluation function this can be guaranteed. For example, if $F(x)$ is convex, the algorithm finds the global minimum. However, in many cases there is no guarantee that the sequence converges to either a local or a global minimum.

Hill Climbing and Simulated Annealing

Hill climbing is an iterative search algorithm that starts with an arbitrary solution. Then it attempts to find a better solution by incrementally changing a single element of the solution. If the change leads to a better solution, an incremental change is made to the new solution. This procedure is repeated until no further improvements can be found.

The search algorithm starts with a guess x, where x is a vector of continuous or discrete values. Each iteration, a single element in x is adjusted and with the adjusted solution x', $F(x')$ is evaluated. If $F(x') \geq F(x)$, the process is repeated with the adjusted solution x' as starting point. The process continues until no change can be found to improve the value of F. Note that the process differs from gradient descent methods, which at each

iteration adjust all values in x according to the gradient at the evaluated point. With hill climbing, one element is adjusted per iteration.

Hill climbing is good for finding local optima or minima, but is not guaranteed to find the global optimum. Moreover, standard hill climbing is not able to escape from local optima, but more advanced variants exist that have the property to sometimes by-pass local optima or minima. One of these variants is simulated annealing. With simulated annealing the probability of making the transition from the current state x to a candidate new state x' is specified by an acceptance probability function $P(F(x), F(x'), n)$. The acceptance probability function depends on the improvement of x', which can be calculated by $F(x') - F(x)$, and the time parameter n. Even if $F(x')$ is worse than $F(x)$, $P\left(F(x), F(x'), n\right)$ is positive. This prevents the method from getting stuck at a local optimum or minimum. Moreover, if the time parameter n increases, the probability $P(F(x), F(x'), n)$ becomes smaller if $F(x')$ is worse than $F(x)$, such that the process ultimately converges to an optimum. Hopefully this optimum is a better optimum than the optimum it would have found with standard hill climbing.

4.2.2.3 Evolutionary Algorithms (EAs)

Evolutionary algorithms refer to a class of search algorithms that are based on Darwinian principles of natural selection and molecular genetics. There are many variants of evolutionary algorithms but the basic idea behind the different variants is the same. In a population of individuals, the individuals that fit best in their environment survive and create offspring. This causes a rise in the fitness of the population. Within this metaphor, the individuals are solutions to the problem at hand, and the offspring are combinations of the naturally selected solutions. In addition, to keep diversity in the population, individuals are subject to random mutations. Figure 4.2 shows the pseudo code of the general evolutionary algorithm and Figure 4.3 shows this scheme as a flow chart. The figures are both from the book of A.E. Eiben and J.E. Smith [8].

The selection procedure for the solutions to be evaluated in the next generation consists of three components:

- Parent selection
- Recombination and mutation
- Survivor selection.

We briefly describe the role of each component. The descriptions are based on the book of Eiben and Smith [8].

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

FIGURE 4.2: From [8]. The general scheme of an evolutionary algorithm in pseudocode

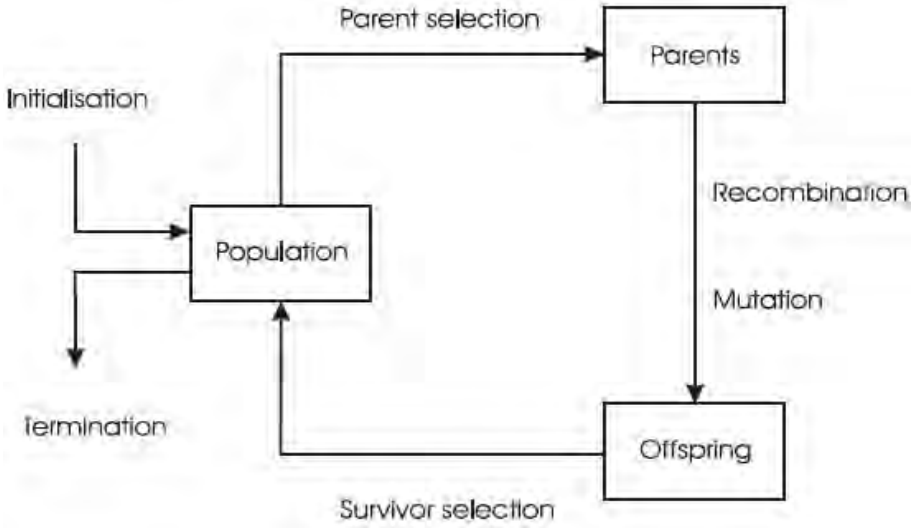

FIGURE 4.3: From [8]. The general scheme of an evolutionary algorithm as a flowchart

Parent Selection

The role of parent selection is to select which individuals become parent of the next generation. An individual is a parent if it has been selected to undergo variation in order to create offspring. Parent selection is typically a stochastic operation, where individuals with a high evaluation score have more chance of becoming a parent than those with a low evaluation score. Nevertheless, to prevent the search from becoming stuck in a local optimum, low-quality individuals are often given a small chance to become parent.

Recombination and Mutation

Recombination and mutation operators have the task to create new individuals from old ones. Recombination merges information from two parents into one or more children, the offspring. Recombination is a stochastic operation. The parts of the parents that are combined and how this is done depend on random drawings. The role of recombination is to combine the properties of two good, but different individuals, such that they produce offspring with a combination of the desirable properties of their parents. Although most of the offspring may not be better or worse than their ancestors, some individuals will combine the effective properties of their parents and will have improved characteristics.

Mutation is applied to one individual and delivers a slightly modified mutant. Mutation is even as recombination a stochastic operation that depends on a series of random choices. By applying mutation to a number of individuals in the population, the population stays diverse and can always escape from local optima. Moreover, mutation assures that in principal all candidate individuals can be reached.

Survivor selection

The final component of the selection procedure is the selection of the survivors after the offspring is produced via recombinations and mutations. In evolutionary algorithms, the size of the population is generally constant, so a choice must be made about which individuals will form the next generation. This is naturally done based on their evaluation score. However, many variants exist where the survivors are selected based on other features, such as age. Survivor selection is in most cases deterministic and the individuals with the highest evaluation score are selected for the next generation.

4.2.3 EA for Unwinding Swaptions

In Subsection 4.2.1 we presented the basic structure of our algorithm to find a good hedge portfolio of swaptions. The first part of the algorithm tries to improve the portfolio by unwinding swaptions that are currently in the portfolio. So it looks for a subset within a set of swaptions that minimizes the objective function (4.1). In this section we describe the Evolutionary Algorithm (EA) that we use to search for this subset.

Representation

The first step in the design of an EA is to choose an appropriate representation of an individual. We are looking for a subset of swaptions within the current set of swaptions

in the portfolio. So the search space consists of all possible subsets. Given a subset of the swaptions in the current portfolio, each swaption of the current portfolio is either in or out of the subset. Therefore, a bit string with the size of the current portfolio is an appropriate representation of an individual. The translation of the bit string to the belonging solution is trivial. Denote the swaptions in the current portfolio with $S_1, S_2, S_3, \ldots, S_N$, where N is the total number of swaptions in the current portfolio. For each $n = 1, 2, 3, \ldots, N$. If the $n-th$ element of the bit string is 1, swaption S_n is in the subset, if the $n-th$ element of the bit string is 0, S_n is not in the subset. Note that the search space grows exponentially and consists of 2^N candidate solutions. With 20 swaptions there are already more than a million candidate solutions, so the brute force method of testing all of i combinations is in most cases not an option.

Initialisation

Initialising an EA means generating the initial population of individuals. Since EAs make in general very quick progression in the first phase, it is often not worth the effort to search for good starting solutions. Instead, the population is filled with random generated individuals, which is also the case in our EA. The population size depends on the number of swaptions in the current portfolio and limitations in the computing capacity. If the portfolio contains a large number of swaptions, the population must also be larger. Otherwise, because the search space increases exponentially, the chance that the algorithm finds a good "neighbourhood" in the search space becomes too small.

Parent Selection

For the selection of the individuals that become parents, we use a tournament selection procedure. With tournament selection, random groups of individuals are composed, the tournaments. Within each tournament, the best individuals are selected to become parent. An alternative choice could have been to use fitness proportional selection. With this type of selection the chance to become selected is proportional to the fitness of the individual. Because our objective function consists of a weighted average of four performance measures, of which three are also averages, the fitness values lie close to each other. This causes the fitness proportional procedure to quite randomly select parents, as the chance for an individual of being selected is proportional to the fitness of that individual. So to keep the selection pressure high, we chose for tournament selection, which not has the drawbacks of fitness proportional selection.

Recombination

Two selected parents are recombined with a one-point cross over operator. This operator chooses a random point in the range $\{1, 2, \ldots, N-1\}$, with N the length of the bit string representing an individual. Then it creates two children by splitting both parents at this point and exchanging the tails, see Figure 4.4.

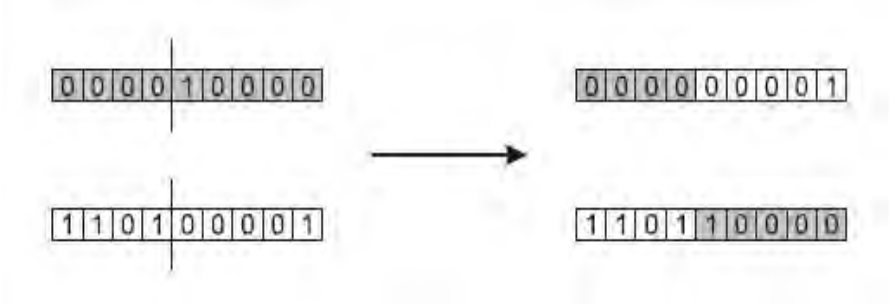

FIGURE 4.4: From [8]. One-point crossover

Mutation

After the children are produced by the recombination operator, they are slightly modified by the mutation operator. The mutation operator works by flipping each bit in the bit string with chance p_m. This chance is also called the bitwise mutation rate. Figure 4.5 shows an example of a mutation for the case where the third, fourth, and eighth elements were selected to flip.

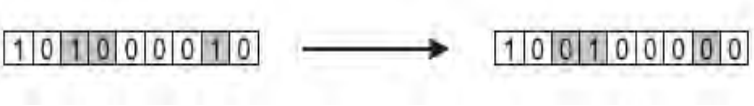

FIGURE 4.5: From [8]. Bitwise mutation

Survival Selection

The individuals that survive are chosen based on their fitness. We place the newly produced individuals in an array in random order. For the current population we do the same. For each index, the best individual of the two individuals at that index in the two arrays survives. This system is illustrated in Table 4.1. The numbers in the table are fictive numbers. The fitness of an individual in our algorithm is determined by the objective function (4.1), that we described earlier.

Old	Fitness	New	Fitness	Survivors	Fitness
Old1	8	New1	9	New1	9
Old2	3	New2	5	Old2	5
Old3	8	New3	8	New3	8
Old4	7	New4	9	New4	9

TABLE 4.1: Illustration of the survival selection procedure with 4 individuals

This procedure ensures that the surviving generation is always better than the old population. You can see that not necessarily the best individuals have to survive. In Table 4.1, individual Old4 does not survive, although it is a better individual than Old2, which does survive. If only the best individuals would survive, this could lead to a situation where the algorithm quickly becomes stuck in local optima. Moreover, it could be that a bad individual has some good properties that otherwise get lost for certain.

Temination Condition

What remains is defining the termination condition. We have chosen to let the EA terminate after a predetermined number of iterations (generations). In the first few iterations the fitness of the population increases quickly. Thereafter, the fitness improves more and more slowly. The number of iterations is chosen such that the population will probably not improve much after termination.

4.2.4 EA for Adding Swaptions

In the second part of the basic structure, described in Subsection 4.2.1, the algorithm improves the hedge portfolio by adding new swaptions. It adds one swaption at a time and searches for properties of that swaption, such that the total hedge portfolio minimizes the objective function (4.1). In this section we describe the Evolutionary Algorithm (EA) that we use to search for a swaption with the most favourable combination of properties.

Representation

In contrast with the EA described in Subsection 4.2.3, the choice of how to represent a solution is not so obvious. Swaptions can differ in time to maturity, tenor, strike price, notional and type (payer or receiver). Moreover, you can hold a long position (buying swaptions), or a shortposition (selling swaptions). For more details about swaptions we refer to Chapter 3. Note that we have binary variables, integer variables and continuous variables. The choice for a payer or receiver swaption and selling or buying a swaptions

are binary. The strike, the notional and the time to maturity of a swaption are continuous variables, and the tenor is an integer variable. Another point of attention is that not all swaptions can be bought on the market, so the variables are bounded. For example, with a current forward swap rate of 2.5%, it is unlikely that a swaption with a strike of 10% is available. This is logical, because the chance that the option becomes in the money, is very close to zero, so the swaption would be worthless.

Because not all values for the different properties of the swaptions are allowed, we define for each property a range of possible values of which one value needs to be chosen. Now, a solution can be represented with a bit string. Each element in the bit string represents a possible value of one of the properties. A one means that the concerning value is chosen. The sum of the elements that represent the possible values of one property must be one. Figure 4.6 shows the range of possible values that we have chosen and the representation of a long position in a payer swaption, that expires in 2 years, has a strike of 3.5%, a tenor of 15 years and a notional of $150 \cdot 10^6$.

initialisation

For the same reasons as described in Subsection 4.2.3, we initialize the population with random individuals. Hereby, The constraint must be satisfied that the sum of the elements that represent the possible values of one property must be one.

Parent Selection

For the parent selection procedure we use the same tournement selection scheme as described in Subsection 4.2.3. The argument to use tournement selection is also valid for this EA. Namely, tournement selection is a simple but effective selection procedure that keeps the selection pressure high, although the fitness values lie relatively close to each other.

Recombination

It is clear that with our representation of an individual, we need to design a recombination operator that produces offspring that satisfy the constraint that the sum of the elements that represent one property must be one. Our recombination operator produces one child from 2 parents. For the properties long/short and payer/receiver, the child inherits randomly from one of the two parents. For the properties time to maturity, strike, tenor and notional, the operator works as follows. If the difference in the

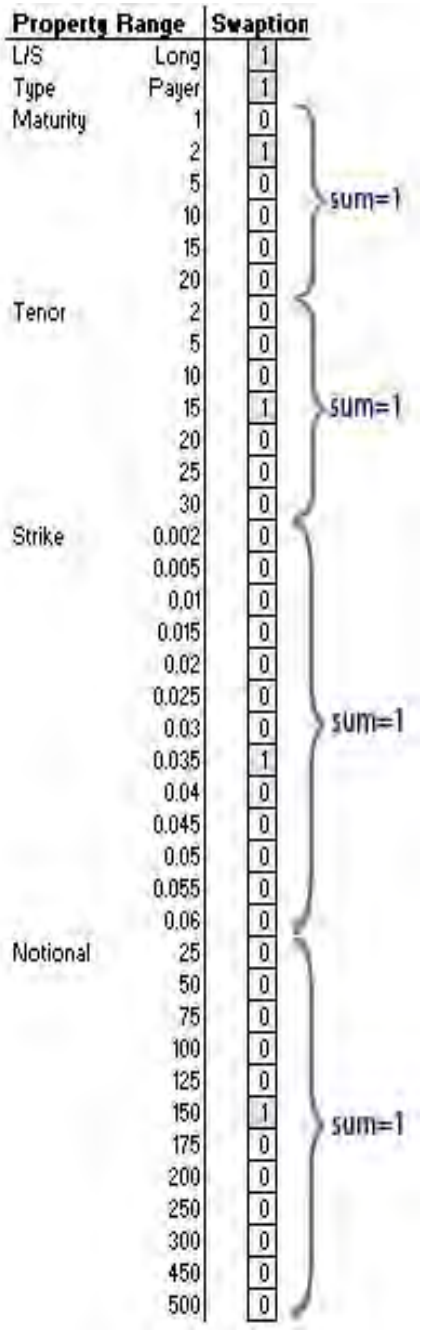

FIGURE 4.6: Representation of individual

indices of the values between the parents is greater than one, the child inherits the value belonging to the average index (converted to the closest integer value). If the difference in the indices is zero or one, the child inherits the property randomly from one of the two parents. Figure 4.7 shows a possible recombination operation.

Property	Range	Parent1	Parent2	Child
L/S	Long	1	0	1
Type	Payer	0	1	1
Maturity	1	0	0	0
	2	1	0	1
	5	0	0	1
	10	0	1	0
	15	0	0	0
	20	0	0	0
Tenor	2	0	0	0
	5	0	0	0
	10	0	0	0
	15	1	0	1
	20	0	1	0
	25	0	0	0
	30	0	0	0
Strike	0.002	0	0	0
	0.005	0	1	0
	0.01	0	0	0
	0.015	0	0	0
	0.02	0	0	1
	0.025	0	0	0
	0.03	0	0	0
	0.035	1	0	1
	0.04	0	0	0
	0.045	0	0	0
	0.05	0	0	0
	0.055	0	0	0
	0.06	0	0	0
Notional	25	0	0	0
	50	0	0	0
	75	0	0	0
	100	0	0	0
	125	0	0	0
	150	1	1	1
	175	0	0	0
	200	0	0	0
	250	0	0	0
	300	0	0	0
	450	0	0	0
	500	0	0	0

FIGURE 4.7: Possible recombination operation

Mutation

Because the recombination operator has a strong averaging character, the mutation operator must be able to cause a large mutation in one or more properties to keep the population diverse. At the same time, the mutation operator must not alter an individual completely, otherwise the new generation is just the result of the selection of random individuals. To achieve this, we use a variant of the bit-wise mutation operator. We change a property of the swaption with chance p_m. If the property is selected for mutation, the property gets a random new value. For the position and type property of a swaption, this means that the bit representing the property will flip. For the other properties, this means that within the range of the bit string that represents the property, a random new element in the range will be one and the others will be zero. Figure 4.8 shows a possible mutation operation where the position and the strike property of the individual are selected for mutation.

Survival Selection and Termination Condition

We use exactly the same procedure for the survival selection procedure and the termination condition as described in Subsection 4.2.3.

Additional Constraints

Sometimes we know something about the solution. For example, for the hedge portfolio of the Unit Linked products, we know that we need receiver swaptions, because the structure of these products is such that SNS REAAL looses money if interest rates decline. To fix a property of an individual, we can simply alter the initialization and mutation operator, such that all individuals in the initial population have a certain property and that this property can not be changed with mutation.

In this chapter we developed a method to find a good hedge portfolio of swaptions to cover the market risks of the embedded options. In the next chapter, we implement this method in Excel with Visual Basics for Applications (VBA). We also present the performance and the results of the algorithm.

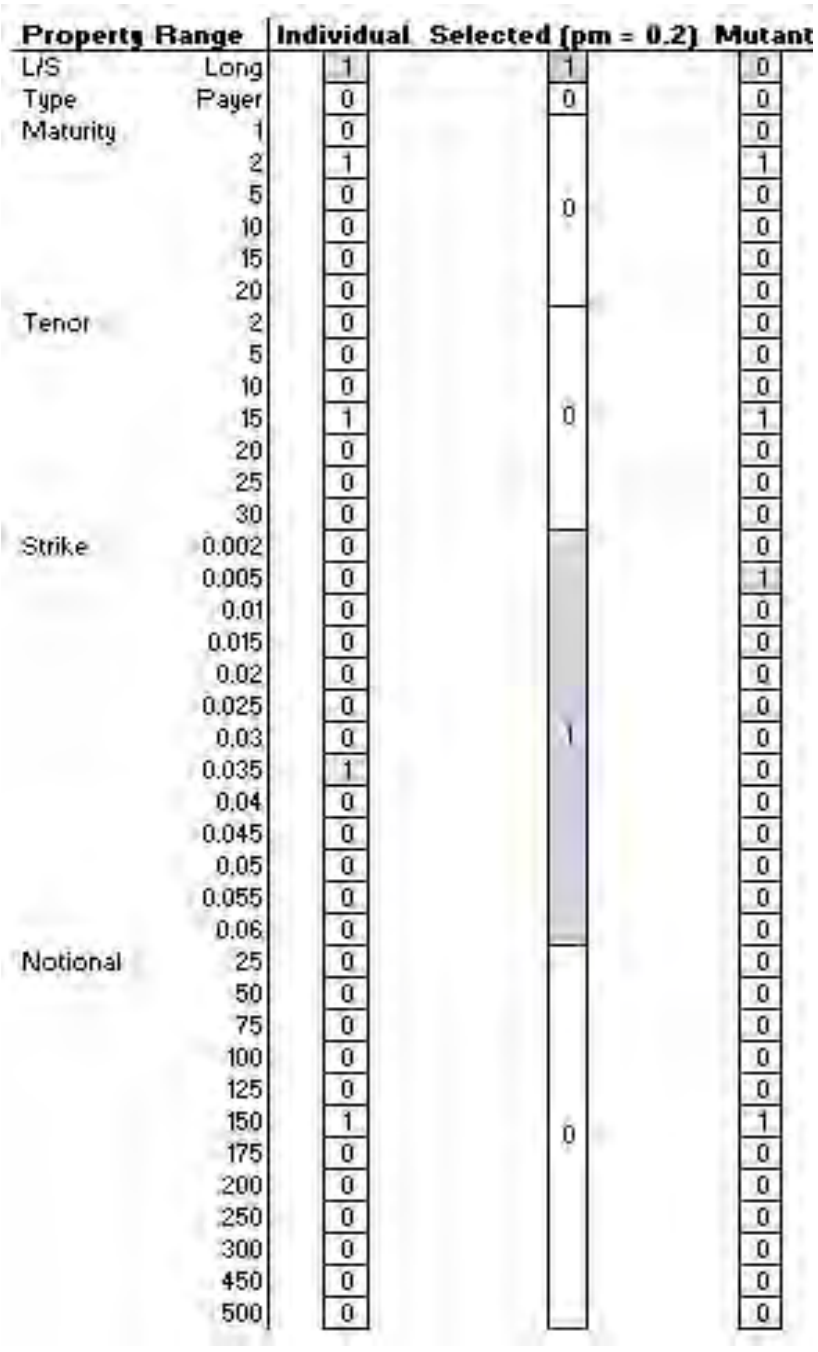

FIGURE 4.8: Possible mutation operation

Chapter 5

Implementing a Hedge Tool in Excel With VBA

5.1 The Worksheets

5.1.1 The Static Data Sheet

5.1.2 The Portfolio Optimization Sheets

5.1.3 A Performance Test for the EA for Unwinding Swaptions

5.1.4 A Performance Test for the EA for Adding Swaptions

5.2 Results

5.2.1 Pension Unit Linked

5.2.2 Pension Direct Profit Sharing

Chapter 6

Reporting on the Hedge Effectiveness

6.1 Measuring Hedge Effectiveness

A measure for the hedge effectiveness quantifies the extent to which changes in the value or cash flows of the hedging instrument offset the changes in the value or cash flows of the hedged item. SNS REAAL is required to perform a retrospective and a prospective hedge effectiveness test to demonstrate that the relationship between the portfolios of swaptions and the hedged portfolios of embedded options has been effective over the last period and is still expected to be effective in the future.

With the prospective test, the expected change of the swaptions is compared with the expected change of the embedded options. The expected changes can be calculated based on the sensitivities of the swaptions, the sensitivities of the embedded options, and one or more interest rate and volatility scenarios. Instead of looking at the expected changes based on sensitivities and interest rate scenarios, we could also look backward at the realized change in value of the embedded options and the realized change in value of the swaptions. This is called retrospective testing.

To quantify either the retrospective or the prospective hedge effectiveness we need a quantitative measure. We describe two commonly used methods, the Dollar Offset Method and Statistical Regression Analysis.

6.1.1 The Dollar Offset Method

The Dollar Offset Method compares the change in value of the hedging instrument with the change in value of the hedged item attributable to the hedged risk. The Dollar Offset measure is defined as the ratio between the change in value of the hedging instrument and the change in value of the hedged item times the hedged risk, where the hedged risk is a percentage of the total market risk of the hedged item. This percentage is called the hedge target. In the case of SNS REAAL, the hedge effectiveness over a certain period is then given by

$$\text{Effectiveness} = \frac{\text{change value swaption portfolio}}{\text{hedge target} \times \text{change value embedded options}}. \tag{6.1}$$

A hedge effectiveness of 100% would mean a perfect hedge portfolio. An effectiveness greater than 100% would mean that the hedge portfolio overreacts and a negative hedge effectiveness would mean that the hedge portfolio and the embedded options react in opposite direction, which is of course undesirable.

A disadvantage of using the Dollar Offset Method is that it gives bad effectiveness results when the changes in value are small. This is referred to as the "the small dollar effect". If, for example, a hedge portfolio of swaptions changes 0.02×10^6 and the hedged portfolio of embedded options changes -0.02×10^6, there is no reason for alarm. Both portfolios have changed very little after all and the change of the swaptions is close to the change of the embedded options. However, based on a hedge target of 50%, your effectiveness measure will indicate a hedge effectiveness of $0.02/(50\% \cdot -0.02) = -200\%$!.

6.1.2 Statistical Regression Analysis

To demonstrate the relationship between the hedging instrument and the hedged item, one could use Statistical Regression Analysis. A number of economical scenarios is generated and the change of the hedging instrument and the hedged item are determined based on these scenarios. Hereafter, a least square fit is applied with the changes of the hedged item as dependent variable and the changes of the hedging instrument as independent variable. The hedge effectiveness is measured based on the slope of the regression line β and the squared coefficient of correlation R^2. β and R^2 must be both as close as possible to 1.

Using this method it is important to ensure that the regression analysis is reasonable and statistically significant. One problem in this respect is establishing an appropriate number of data points to regress. Generally one needs at least 30 data points, which means that at least 30 economical scenarios must be considered.

An advantage of using Statistical Regression Analysis to asses the hedge effectiveness, is that the problem of small numbers is not present. However, the method generally costs more effort to apply than the Dollar Offset Method. Moreover, the Dollar Offset Method delivers one intuitive measure for the hedge effectiveness, while the regression method produces two measures, β and R^2, which are less intuitive for outsiders.

6.2 Clarification of the Hedge Effectiveness

6.2.1 Reconciliation of the Change of the Swaptions

Example of the Reconciliation Process

6.2.2 Reconciliation of the Change of the Embedded Options

6.3 The Report

6.3.1 Input Parameters

6.3.2 Updating the Data

6.3.3 The reports

Chapter 7

Discussion and Conclusions

Appendix A

Screenshots of the Hedge Effectiveness Report

Bibliography

[1] http://nl.wikipedia.org/wiki/simulated_annealing.

[2] http://en.wikipedia.org/wiki/brute-force_search.

[3] http://en.wikipedia.org/wiki/hill_climbing.

[4] http://en.wikipedia.org/wiki/search_algorithm.

[5] http://en.wikipedia.org/wiki/steepest_descent.

[6] ALM-V. *Modelbeschrijving ALM tool*. SNS REAAL, 2013.

[7] T. Bjork. *Arbitrage Theory in Continuous Time*. Oxford University Press, 2009.

[8] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2007.

[9] Team Opties en Garanties. *Methodologie document Analytische waardering Opties en Garanties*. SNS REAAL, 2013.

[10] KPMG. *Basics of Hedge Effectiveness Testing and Measurement*. FINCAD, http://www.cmegroup.com/education/files/basics-of-hedge-effectiveness.pdf, 2011.

[11] D.C. Lay. *Linear Algebra And Its Applications*. Pearson Eduction US, 2006.